الطبعة الثانية

...ب الكتروني وقواعد البيانات

DBASE III +

FOXBASE +

FOXPRO

DBASE IV

د. جمال عبد المعطي — ...رضا عبد الوهاب

أ.د. محمد على الشرقاوي — ...الدين محمد فهمي

د. عزت إبراهيم شـداد — ...عزيز حسن الحيري

تحقيق وتقديم

أ.د. محمد فهمي طلبه

٥

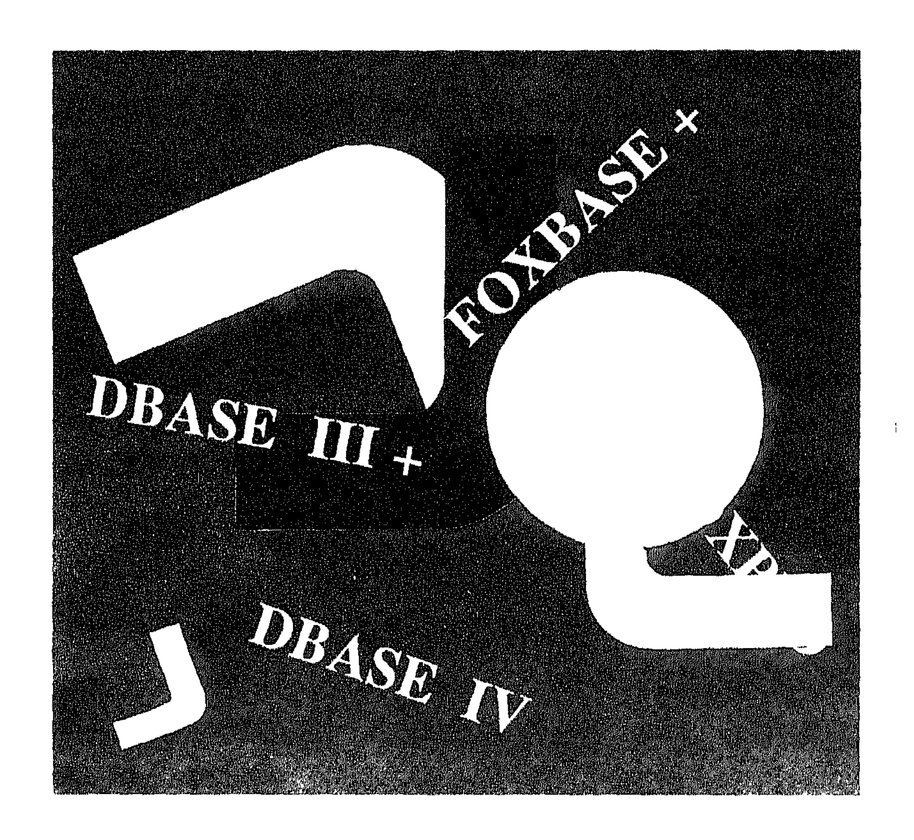
مجموعة كتب دلتا



الحاسب الإلكترونى
وقواعـــد البيـــانـــات



الحاسب الإلكترونى وقواعـــد البيـــانـــات



م.مصطفى رضا عبد الوهاب — د.جمـــال عبـــد المعطـــى
د. عـــلاء الديـن محمـد فهمـى — ا.د. محمـد علـى الشرقـــاوى
م. عبـد العـزيـز حسـن الحريرى — د. عـــزت إبراهيـم شـــداد

تحقيق وتقديم

ا.د. محمـــد فهمـــى طلبـــه

٥

مجموعة كتب دلتـــا

رقم الإيداع ٩٠١٩/٩٣



ت: ٢٢١١٠٧١-٢٢١١٠٧٢

تقـــديـــم

إن التطور فى لغات برمجة الحاسب قد مر بمراحل متعددة إرتبطت بعدة عوامل من أهمها التطور التكنولوجى المتلاحق فى مكونات الحاسب المادية وماتبع ذلك من زيادة سرعة عمليات الحاسب المختلفة وزيادة كفاءتها. وقد أدى هذا إلى تطور هائل فى نظم التشغيل وظهور مفاهيم جديدة مثل تعدد الوظائف وتعدد المستخدمين واستخدام الشبكات بالإضافة إلى العديد من الخصائص التى تميز الأجيال المتلاحقة من الحاسب الإلكترونى.

ومن الملامح الرئيسية لتطور لغات البرمجة البعد التدريجى عن التدخل فى العديد من العمليات التفصيلية التى تتم بواسطة الحاسب لأداء مهمة معينة مما أتاح الفرصة أمام مخططى البرامج لاستخدام خصائص الحاسب دون الحاجة إلى الإلمام الكامل بعملياته الداخلية. ومن المعروف أن الحاسب حقيقة لايفهم إلا لغة " الواحد " و " الصفر " وهى مايطلق عليها لغة الماكينة (Machine Language). وكان مخطط البرامج لايستطيع التفاهم مع الحاسب إلا من خلال هذه اللغة المعقدة. ثم ظهرت لغات أكثر سهولة قامت بتكوين مجموعات من الواحد والصفر فى رموز بسيطة واستخدام هذه الرموز فى كتابة البرامج. وتسمى هذه اللغات باللغات الرمزية (Symbolic Languages) أو لغات التجميع (Assembly Languages). وأخذت اللغات فى التطور مع زيادة درجة التمثيل حتى وصلت الآن إلى لغات الجيل الرابع (4th Generation Languages) حيث أتاحت هذه اللغات لمخطط البرامج التعامل مع النوافذ والقوائم الواضحة التى يستطيع من خلالها تصميم البرامج المطلوبة بسهولة تامة ويسر.

وقد إنعكس هذا التطور الكبير على المتخصصين فى مجال الحاسب حيث أصبح على مخططى البرامج متابعة كل جديد فى مجال نظم تطوير البرامج وأدواتها المتقدمة وذلك حتى يمكنهم الإستفادة من خصائصها فى تصميم النظم المتميزة التى توفر الكفاءة العالية وسهولة الإستخدام. ومن ناحية أخرى فقد أصبح على المستخدم ضرورة الإلمام بهذه البرامج التطبيقية الحديثة حتى يستطيع الإعتماد على نفسه فى

الإستفادة منها والإضافة إليها. وقد أدى ذلك إلى ظهور جيل جديد من المستخدمين الذين يمتلكون خبرة كبيرة فى التعامل مع العديد من البرامج التطبيقية إلى جانب القدرة على تصميم النظم الخاصة بها.

وقد كان لهذا التطور فى مجال الحاسبات أثره فى طبيعة الكتب المتخصصة على المستوى العالمى فى هذا المجال. فبعد أن كانت هذه الكتب - إلى وقت قليل مضى - تركز على الجوانب النظرية ، أصبحت الآن تركز على أساليب إستخدام التطبيقات وعلى تقديم الخبرات والمهارات العملية لمستخدمى الحاسبات.

وقيام مؤسسة " دلتـــا" بتقديم مجموعتها الجديدة لتكنولوجيا وعلوم الحاسب يعوض النقص الشديد الذى تعانى منها المكتبة العربية فى هذا المجال حيث أن معظم الكتب العربية الموجودة ليست سوى ترجمة أو تلخيص سطحى لدليل التشغيل لنظم الحاسب المختلفة بينما يحتاج المستخدمة إلى توضيح الكثير من الجوانب العلمية والفنية بالإضافة إلى خصائص تشغيل النظم. وهذا الجهد ماهو إلا إمتداد لسياستها الواعية وإحساسها بمسئوليتها نحو التطور التكنولوجى بالمنطقة العربية.

وهذا الكتاب هو أحد كتب " مجموعة كتب دلتا " وهو يمثل حلقة الإتصال بين الجوانب التطبيقية والجوانب العلمية والفنية فهو يبدأ فى الجزء الأول بشرح المفاهيم الأساسية المرتبطة بقواعد البيانات متضمنة تمثيل البيانات وتراكيب البيانات وعمليات الفرز والبحث ومراحل تصميم قاعدة البيانات ثم يشرح لغة (SQL) كنموذج لقواعد البيانات العلاقية ونظام (CODASYL) كنموذج لقواعد البيانات الهرمية ونظم (IDMS) كنموذج لقواعد البيانات الشبكية ثم ينتقل إلى شرح نظم إدارة قواعد البيانات. ثم يقدم الكتاب فى الجزء الثانى شرحا دقيقا وشاملا لبرامج عائلة (DBase) التى تشمل (DBaseII) ، (DBaseIII) ، (DBaseIV)، (Clipper) ، (Foxbase) ، (FoxPro) وذلك كنموذج لنظم إدارة قواعد البيانات. كما يتضمن الجزء الثالث أهم الأوامر المستخدمة فى برامج عائلة (DBase). وفى الجزء الرابع يشرح الكتاب أساسيات تحليل وتصميم النظم وأدوات الـ (CASE) بالإضافة إلى شرح أحد منهجيات الـ (CASE) المعروفة وهى منهجية (HOS).

والكتاب فى مجمله يوفر الإحاطة الكاملة بكل مايتعلق باستخدام قواعد البيانات لبناء نظم الحاسب. كما أنه مقسم بطريقة تركيبية بحيث يستطيع كل قارىء إختيار الأجزاء التى تناسبه طبقا لمعلوماته وإحتياجاته. فالقارىء الذى يريد الإلمام بالجوانب النظرية يمكنه البدء من الجزء الأول. والقارىء الذى يهتم فقط بالجوانب التطبيقية يمكنه القفز مباشرة إلى الجزء الثانى. أما القارىء الذى يريد الإحاطة بأساسيات تحليل النظم وأدوات الـ (CASE) فيمكنه الإنتقال مباشرة إلى الجزء الرابع.

والله الموفق ،،،

ا . د . محمد فهمى طلبه



محتويات الكتاب

مسلسل	الموضوع	رقم الصفحة

مسلسل	الموضوع	رقم الصفحة

مسلسل　　　　　　　　　　　　الموضوع　　　　　　　　　　　　رقم الصفحة

مسلسل	الموضوع	رقم الصفحة

الفصل الخامس عشر " تعديل السجلات (Updating Records) " ---- ٢٤١

الفصل السادس عشر " تنظيم الملف (File Organiztion) " ------------ ٢٥٣

مسلسل	الموضوع	رقم الصفحة

مسلسل	الموضوع	رقم الصفحة

مسلسل الموضوع رقم الصفحة

مسلسل الموضوع رقم الصفحة

مسلسل	الموضوع	رقم الصفحة

مسلسل	الموضوع	رقم الصفحة

مسلسل	الموضوع	رقم الصفحة



الجزء الأول

قواعـــد البيـانـات



مقـــدمـــة

يوضح هذا الجزء مفهوم قواعد البيانات من زاوية أكاديمية تعطى القارىء رؤية واضحة تعينه على التعامل مع مختلف النظم والتطبيقات المبنية على قواعد البيانات. ويغطى هذا الجزء مختلف المواضيع المرتبطة بقواعد البيانات متضمنا الشرح الدقيق والواضح والمصحوب بالأمثلة العملية من البيئة الفعلية. وقد تم إعداد هذا الجزء بحيث يكون مستقلا عن الأجزاء التالية بما يتيح للقارىء الذى لايريد الدخول بعمق فى الدراسة الأكاديمية لقواعد البيانات ويريد الدخول مباشرة فى الدراسة التكنولوچية والتطبيقية لأهم نظم إدارة قواعد البيانات المعروفة أن يتخطى هذا الجزء وينتقل مباشرة إلى الجزئين الثانى والثالث.

ويتضمن هذا الجزء شرح المفاهيم الأساسية لقواعد البيانات وطرق تمثل البيانات وتراكيب البيانات ووسائل الفرز والبحث ثم ينتقل إلى تصميم قواعد البيانات وأنواع قواعد البيانات المختلفة مع شرح لغة (SQL) كنموذج لقواعد البيانات العلاقية ونظام (CODASYL) كنموذج لقواعد البيانات الهرمية ونظام (IDMS) كنموذج لقواعد البيانات الشبكية. أما القارىء الذى يريد الإطلاع على معلومات أكثر عمقا عن هذا الموضوع فيمكنه بعد قراءة هذا الجزء الإنتقال إلى الجزء الرابع الذى يوضح أساسيات تحليل وتصميم النظم كما يشرح أدوات الـ (CASE) التى تتيح للمستخدم تصميم نظم الحاسب دون كتابة أى كود.



الفصل الأول

مفاهيم أساسية



١ - ١ البيانات والمعلومات (Data and Informations)

البيانات (Data) هى مجموعة من الحقائق (Facts) التى تعبر عن مواقف وأفعال معينة وهى ما يطلق عليها (Entities) أى كيانات مستقلة ويتم التعبير عنها بالكميات أو الرموز أو الأرقام. وهناك مصادر متعددة لهذه البيانات داخل أى تنظيم مثل بيانات المبيعات والمرتبات والبيانات الشخصية ... إلخ. ويجب أن نفرق بين البيانات والمعلومات. فالمعلومات (Information) هى بيانات يتم تجميعها وتنظيمها ومعالجتها وتحليلها لتعطى دلالة معينة تساعد على إتخاذ القرار. أى يمكن القول أن البيانات هى المادة الخام والمعلومات هى مانستخلصه من البيانات بعد معالجتها. فمثلا قائمة أسماء الطلبة فى فصل هى فى الواقع بيانات ، فى حين تصبح هذه البيانات معلومات عند ترتيب هذه الأسماء حسب متوسط درجات كل طالب من الأعلى إلى الأقل لتحديد الترتيب العام لهؤلاء الطلاب.

وكل وحدة بيانات (Data Item) لها خواص معينة تشمل إسم هذه الوحدة (Data Item Name) وطولها (Length) ونوعها (Type) وقيمتها (Value). ويساعد إسم وحدة البيانات على سهولة تمييزها عن باقى وحدات البيانات وطول وحدة البيانات يستدل منه على عدد الحروف أو الأرقام المستخدمة فى تكوينها. أما نوع وحدة البيانات فيحدد ما إذا كانت هذه الوحدة حرفية (Alphabetic) أو عددية (Numeric) أو حرفية عددية (Alphnumeric).

والبيانات تحتاج عادة إلى تسجيلها أو نقلها حسب درجة أهميتها. ويتم التسجيل بإستخدام وحدات البيانات الثنائية (Bits) التى تتكون من الرقم (0) والرقم (1). وتمثيل البيانات (Modeling) يستخدم نماذج بيانات (Data models) تمثل حلقة الإتصال بين العالم الحقيقى (Real world) والحاسب.

١ - ٢ نماذج البيانات (Data Models)

نماذج البيانات هى وسيلة أو أداة (Tool) تستخدم لتمثيل البيانات بصورة يسهل إستخدامها بواسطة الحاسب. وبمعنى أكثر شمولا فإن نماذج البيانات توضح معنى البيانات والعلاقات التى تربط بينها فيما يعرف بتراكيب البيانات (Data Structures) كذلك العمليات المسموح بها على هذه التراكيب.

نمــاذج البيــانات

هى نماذج توضح القواعد العامة لتراكيـب البيـانات (Data Structures) والعمليات المسموح بها على هذه التراكيب.

ومن أهم نماذج البيانات التى تستخدم بصفة خاصة مع قواعد البيانات هى مايطلق عليه المخطط (Schema). وهذا المخطط يوضح الكيانات (Entities) التى يتم تمثيلها مثل الموظفين أو الطلبة أو قطع الغيار ... إلخ والتى يطلق عليها أيضا الصنف (Category) وكذلك خصائص هذه الكيانات (Properties) (مثل إسم الموظف ، إسم الطالب ، رقم الجزء ... إلخ) بالإضافة إلى العلاقات (Relations) بين هذه الكيانات.

ومن نماذج البيانات التى نقابلها فى الحياة العملية نموذج التقدم لوظيفة أنظر شكل (١ - ١).

طلب توظيف

الوظيفة : .... **الرقم الكودى :** ....

(١) البيانات الشخصية

الإسم : .... **تاريخ الميلاد :** .... **محل الميلاد :** ....

الجنسية : .... **الديـــانة :** .... **الحالة الإجتماعية :** ....

محل الإقامة : .... **الموقف من التجنيد :** .... **رقم التليفون :** ....

البطاقة (شخصية/عائلية) : .... **رقمها :** .... **جهة وتاريخ إصدارها :** ....

(٢) المؤهل الدراسى : .... **تاريخ الحصول عليه :** .... **التقدير :** ....

(٣) الخبرات السابقة :

....

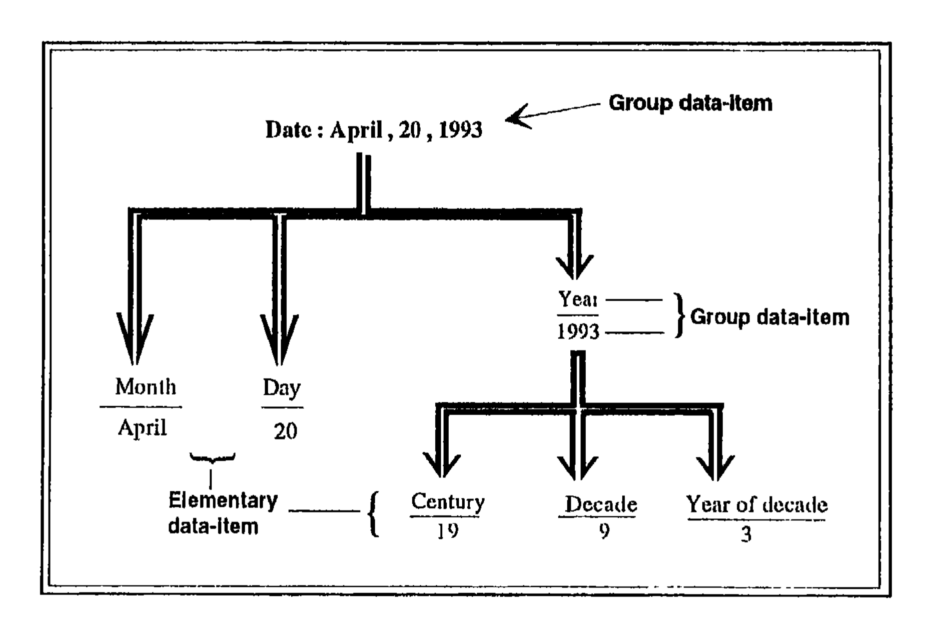
شكل (١ - ١)

ويلاحظ أن البنية الأساسية لتكوين أى نموذج بيانات هى وحدة البيانات (Data Item) (الإسم ، العنوان ، رقم التلفيون ... إلخ) ويطلق إسم (Context) على أسماء وحدات البيانات فى حين يطلق على القيمة الخاصة بها (Value) والجدول التالى يوضح أسماء وحدات البيانات والقيم الخاصة بها.

Data Item Name	Data Item Value
Name	Ahmed Salem
Sex	Male
Address	30 - Ain Shams - Cairo
TEL	271365

١ - ٣ تجميع وتجزئة البيانات

هناك بعض وحدات البيانات تتكون من وحدات بيانات أصغر مرتبطة ببعضها فيما يسمى (Group Data Items). فمثلا التاريخ عادة يحتوى على ثلاثة أجزاء (اليوم ، الشهر ، السنة) والسنة بدورها يمكن تقسيمها إلى ثلاثة أجزاء (القرن (Century) العقد (Decode) ، السنة الحالية).



شكل (١ - ٢)

وهناك بعض وحدات البيانات التى لايمكن تحليها إلى بيانات أصغر وتسمى وحدات بيانات عنصرية (Elementary Data Items) مثل وحدة البيان (Month) فى

شكل (١ - ٢) والذى يوضح التاريخ (Date) كأحد وحدات البيانات المجمعة وكذلك الشكل الهرمى الذى يوضح تحليل هذه البيانات إلى وحدات أصغر.

وتتعدد العمليات التى تعتمد على وحدات البيانات العنصرية مثل عمليات البحث (Searching) والتعديل (Editing) والتحديث (Updating) بالإضافة إلى العمليات التى تعتمد على وحدات البيانات المجمعة مثل الفرز (Sorting) والتجميع (Batching) والترتيب (Ordering).

١ - ٤ السجـــلات (Records)

السجل هو تجميع لوحدات البيانات المتعلقة بكيان معين (Entity) سواء كان شخصا أو حدثا أو شيئا. وهذه البيانات تكون مرتبطة ببعضها مما يسهل التعامل معها ومعالجتها (Data Processing) ككيان مستقل بمعنى آخر فإن السجل هو تجميع لوحدات البيانات ذات الإرتباط المشترك مرتبة ومنظمة بطريقة تساعد الحاسب على التعرف عليها والتعامل معها من خلال تسميتها بإسم وصفى (Descriptive Name) لاستخدامه من خلال برنامج معين للرجوع إلى السجل.

فمثلا السجل الخاص بكيان الموظفين نستطيع أن نسميه (EMPLOYEE) ويمكن أن يحتوى على البيانات الموضحة فى الجدول التالى :

RECORD : EMPLOYEE

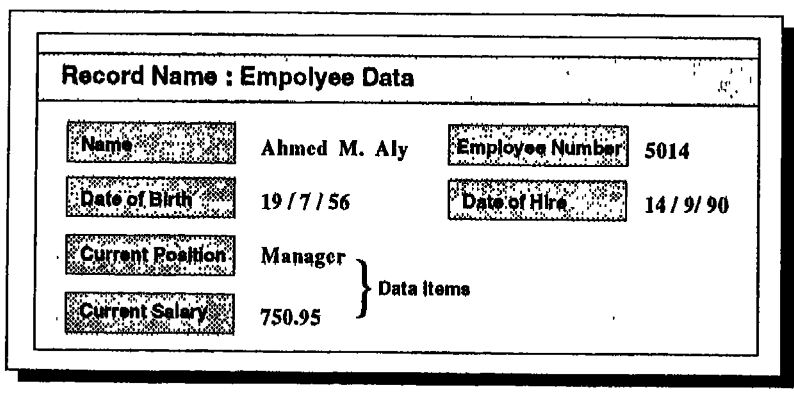
Data Item Name	Length	Type	Value
EMPOLYEE NAME	10	A (i.e. Alphabetic)	Aly Hassan
INDENTIFICATION NUMBER	6	N (i.e. Numeric)	343675
ADDRESS	20	X	4-Ahmed Maher St., Roxy
CITY	16	X	CAIRO

مما سبق يتضح أن السجل يجب أن يكون له شكل ونظام محدد (Format) وهذا الشكل يجب أن يحتوى على المكونات الآتية :

١ - أسماء وحدات البيانات (Data Items Names).
٢ - طول كل وحدة بيانات (Length).
٣ - نوع وحدة البيانات (Type) ، وهى أما
- عددية (Numeric).
أو - حرفية (Alphabetic).
أو - حرفية عددية (Alphnumeric).
أو - تاريخ (Data).
٤ - القيم المسموح بها فى وحدة البيانات (Value).

١ - ٥ الحقـــول (Fields)

الحقول هى التمثيل الفعلى لوحدات البيانات (Physical Representation) داخل الحاسب. والحقل هو مجموعة من الحروف تحتل مكانا محددا داخل السجل. فمثلا يحتوى سجل أى موظف على بعض المعلومات المحددة مثل الإسم ، السن ، تاريخ التعيين ... إلخ ، وتسمى هذه الوحدات حقول (Fields) والحاسب يقوم باختبار كل حقل والتأكد من مطابقته للشروط المحددة فى البرنامج. فمثلا عند تخصيص حقل للتاريخ يجب تخصيص حرفين لليوم لتغطية الأعداد حتى (٣١) وفى نفس الوقت يجب تحديد الحد الأقصى للأعداد فى هذا الحقل حتى لايسمح بادخال عدد أكبر من (٣١). والشكل (١ - ٣) يوضح بيانات تفصيلية عن سجل أحد الموظفين والحقول المستخدمة فى تكوينه.



شكل (١ - ٣)

والبيانات الخاصة بهذه الحقول يمكن عرضها فى الجدول التالى :

Field Number	Title (Name)	Type	Length
1	Name	A	20
2	Employee Number	N	10
3	Date of birth	D	8
4	Date of hire	D	8
5	Current Position	A	20
6	Current Salary	N	8

A : Alphabetic N : Numeric D : Date

ويلاحظ أن لكل حقل بيانات رقم خاص به وإسمه ونوعه وطوله.

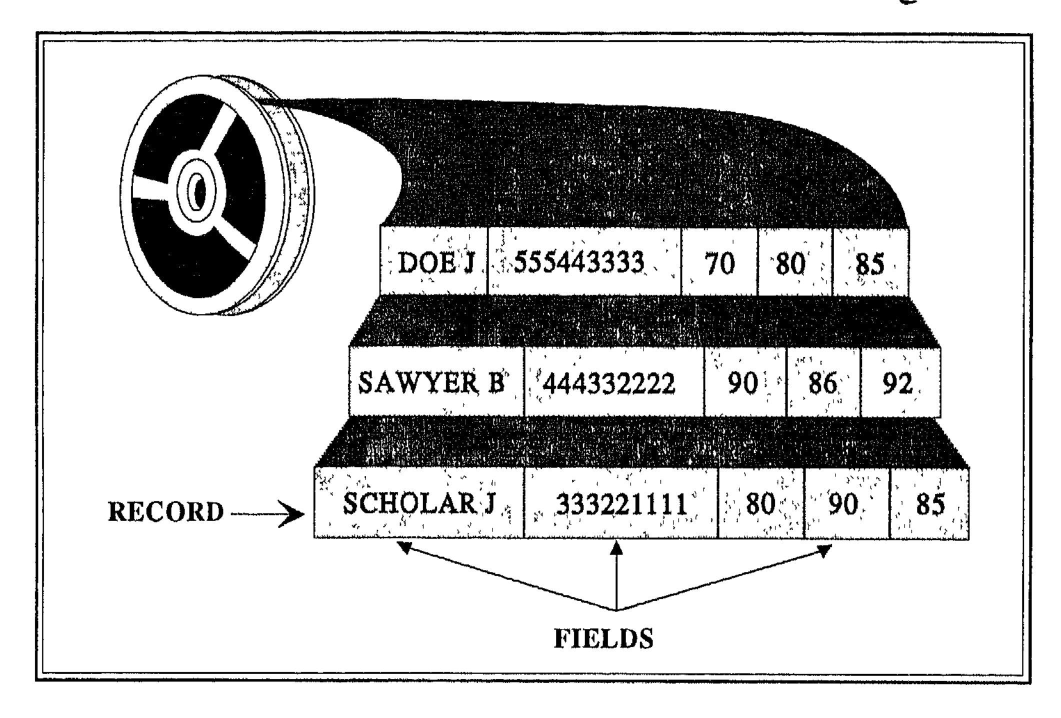
١ - ٦ الملفـــات (Files)

الملف هو تجميع للسجلات الخاصة بموضوع محدد مثل ملف الموظفين(Employees) وملف حسابات العميل (Checking Account File) أو ملف طلبات الشراء (Purchase Order File). وحتى يستطيع الحاسب قراءة الملف وإسترجاع أى سجل فيه فإن تخزينه يتم بطريقة محددة. وهناك طريقتان لتنظيم الملف لتحقيق هذا الهدف. الطريقة الأولى هى إستخدام التشغيل المتتابع للملفات والطريقة الثانية هى إستخدام التشغيل المباشر للملفات.

١ - ٦ - ١ التشغيل المتتابع للملفات (Sequential Access Files)

الملف المتتابع (Sequential File) هو ملف يتم تنظيم السجلات فيه بشكل ثابت ومتتابع كما هو واضح فى شكل (١ - ٤). وعملية التشغيل تعنى استرجاع سجلات الملف بالترتيب الذى تم إدخالها به فى البداية. فمثلا عند استرجاع سجل معين يجب البحث خلال الملف كله من البداية تتابعيا (Sequentially) ومرورا بكل

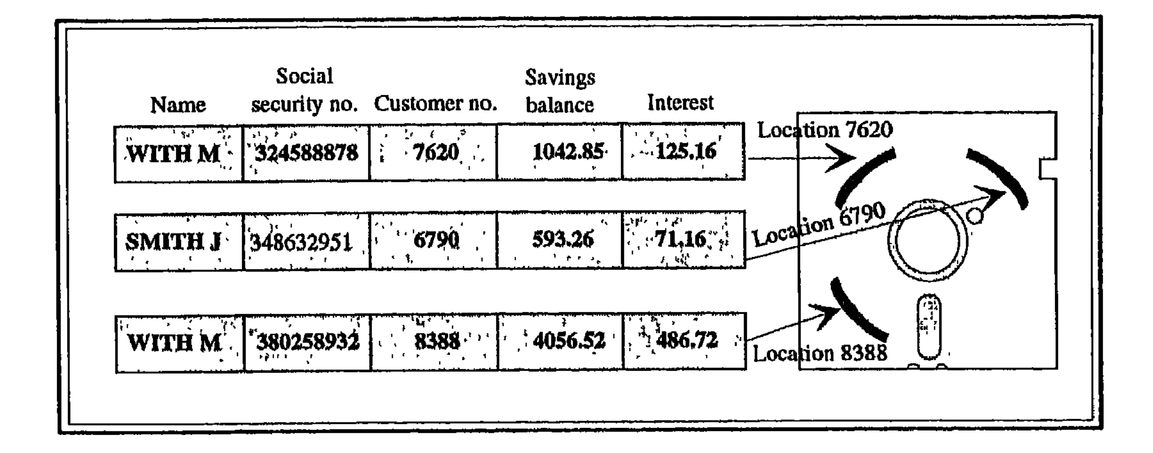
السجلات حتى الوصول إلى السجل المطلوب. كذلك فإن السجلات الجديدة يمكن فقط تخزينها فى آخر الملف. والوسيلة الشائعة للتعامل مع هذا الملف هى تخزينه مرتبا باستخدام مفتاح (Key) معين وليكن إسم الطالب مثلا وباستخدام هذا المفتاح يمكن استرجاع أى سجل.



شكل (١ - ٤)

١ - ٦ - ٢ التشغيل المباشر للملفات (Direct Access Files)

ملف التشغيل المباشر هو ملف يتم تخزين السجلات به طبقا لنظام عنونة (Addressing Scheme) معين كما هو مبين فى شكل (١ - ٥). ويستخدم عنوان السجل (Record Address) فى إسترجاعه مباشرة دون المرور على كل السجلات التى تسبقه. ويمكن أن يكون عنوان السجل هو حقل خاص من حقول السجل أو أحد الحقول المستنتجة. وهذا الحقل يجب أن يكون منفردا (Unique) مثل رقم العميل (Customer No.) أو رقم تحقيق الشخصية ... إلخ. ويتم ذلك باستخدام فهرس يوضح رقم السجل مقابل رقم العميل. وعند استرجاع سجل معين يتم استخدام رقم العميل والفهرس فى تحديد رقم السجل والذى من خلاله يمكن الوصول مباشرة إلى السجل المطلوب. أنظر شكل (١ - ٥).



شكل (١ - ٥)

والتشغيل المباشر للملفات يعتبر أفضل كثيرا من التشغيل المتتابع فى الحالات التى يكون فيها تشغيل الملف قاصرا على التعامل مع سجلات منفردة وليس مع مجموعات متتالية من السجلات. فى حين يكون التشغيل المتتابع أفضل فى الحالات التى يكون فيها التعامل مع مجموعات كبيرة من السجلات التى يتم تشغيلها كمجموعة واحدة. وذلك مثلا عندما يراد كتابة المرتبات لجميع الموظفين بصفة دورية أو طباعة فواتير شهرية للعملاء .... وهكذا.

١ - ٧ قواعد البيانات (Databases)

من سمات العصر الحاضر أن حجم المعلومات قد تضخم بدرجة كبيرة نتيجة التقدم العلمى والتطور التكنولوجى. وهذا الكم الهائل من المعلومات أصبح عنصرا هاما ومؤثرا على جوانب عديدة من المجتمع لذلك أصبح من الضرورة بمكان وجود نظام حاسبات يسمح بتخزين هذه المعلومات فى ملفات مرتبطة منطقيا ومتعلقة بكيان واحد فيما يعرف بقواعد البيانات (Databases). ونظم قواعد البيانات تتيح التعامل مع هذه المعلومات من حيث التخزين والإسترجاع والحذف والإضافة والعرض وذلك بالإضافة إلى إخراجها مطبوعة عند الحاجة وتسمى هذه النظم نظم إدارة قواعد البيانات (Database Management Systems). وذلك يؤدى إلى سرعة ودقة إتخاذ القرارات كما يساعد على التخطيط الشامل للمشروعات. ومن أمثلة هذه النظم نظم المرتبات ونظم حسابات العملاء ونظم السيطرة على المخزون ونظم الحجز فى شركات الطيران ونظم المرضى

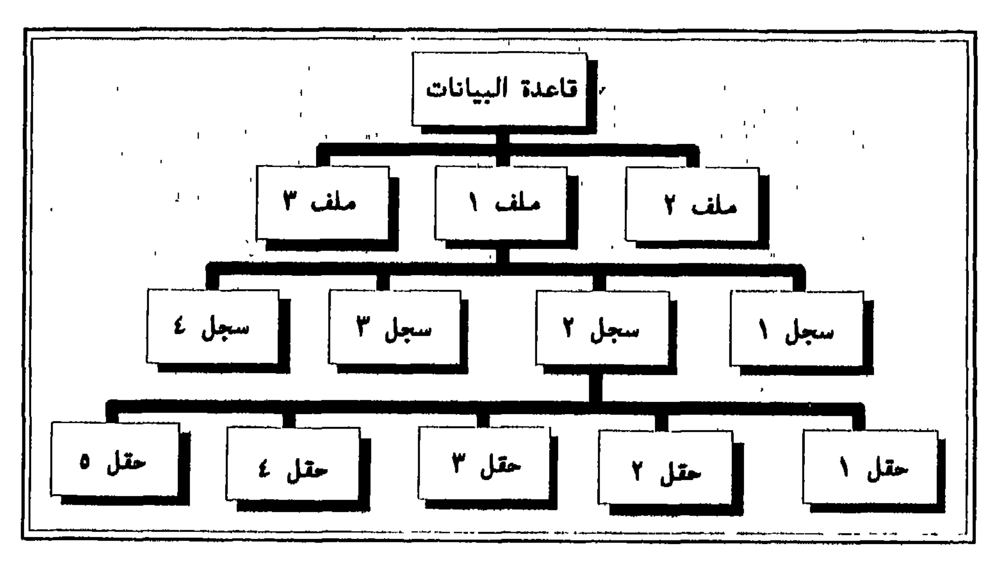
فى المستشفيات ونظم المكتبات و ... إلخ.

> ***قاعدة البيانات***
>
> *هى مجموعة من الملفات المرتبطة منطقيا والمتعلقة بكيان واحد.*

فمثلا هناك إرتباط بين الملف الذى يحتوى على البيانات الشخصية للموظفين (Personnel) وملف بيانات المرتبات (Payroll File) لنفس الموظفين لأن هذين الملفين يحتويان على معلومات عن نفس الكيان وهو الموظفين لذلك فإن كل سجل لموظف فى ملف البيانات الشخصية يقابله سجل آخر لنفس الموظف فى ملف المرتبات. هذه الملفات المرتبطة تكون فى مجموعها قاعدة بيانات.

وتتكون قاعدة البيانات من مجموعة من مستويات البيانات التى يمكن تمثيلها على شكل هرمى وهذه المستويات هى : انظر شكل (١ - ٦).

١ - وحدة البيانات (Data element)
٢ - الحقل (Field).
٣ - السجل (Record).
٤ - الملف (File).



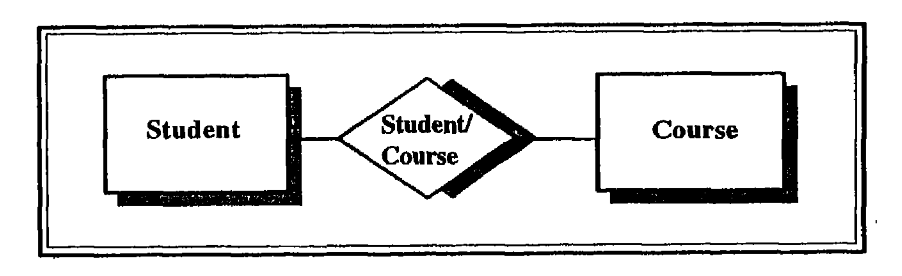
شكل (١ - ٦)

١ - ٨ النماذج المنطقية والفعلية لقواعد البيانات
(Logical and Physical Models for Databases)

عند إنشاء قاعدة البيانات فإن الخطوة الأولى هى إنشاء النموذج المنطقى (Logical Model) لقاعدة البيانات. وهذا النموذج المنطقى هو تمثيل مختصر للكيانات الخاصة بقاعدة البيانات والعلاقات بين هذه الكيانات وذلك دون الدخول فى التفاصيل المتعلقة بالبرامج (Software) والمكونات المادية (Hardware) التى سوف تتعامل مع هذه الكيانات.

والخطوة الثانية لإنشاء قاعدة البيانات هى إنشاء النموذج الفعلى(Physical Model) لقاعدة البيانات وهو النموذج الذى يقوم بتوصيف قاعدة البيانات توصيفا دقيقا متضمنا إسم القاعدة وأسماء السجلات وأسماء الحقول وأنواع الحقول وحجمها ... إلخ.

فمثلا الشكل (١ - ٧) يوضح النموذج المنطقى لقاعدة بيانات إسمها (College) ويمثل الطلبة الملتحقين بدراسات (Courses). حيث يتم تمثيل الكيانات بصناديق (Boxes) والعلاقة بينهما بشكل معين (Diamond) والشكل (١ - ٨) يوضح النموذج الفعلى لنفس قاعدة البيانات حيث يتم تحديد خصائص حقول كل علاقة مثل إسم الحقل ونوعه وعرضه و ... إلخ.



شكل (١ - ٧)

DATABASE NAME COLLEGE

RELATION NAME STUDENT

01	NAME	C	40	
02	SSN	C	11	
.				
.				
.				

RELATION NAME COURSE

01	ID NUMBER	N	5	0
02	TIME	C	5	
03	LOCATION	C	20	
.				
.				
.				

RELATION NAME STUDENT/COURSE

01	SS	C	11	
02	ID NUMBER	N	5	0

شكل (١ - ٨)



الفصل الثاني

تمثيل البيانات

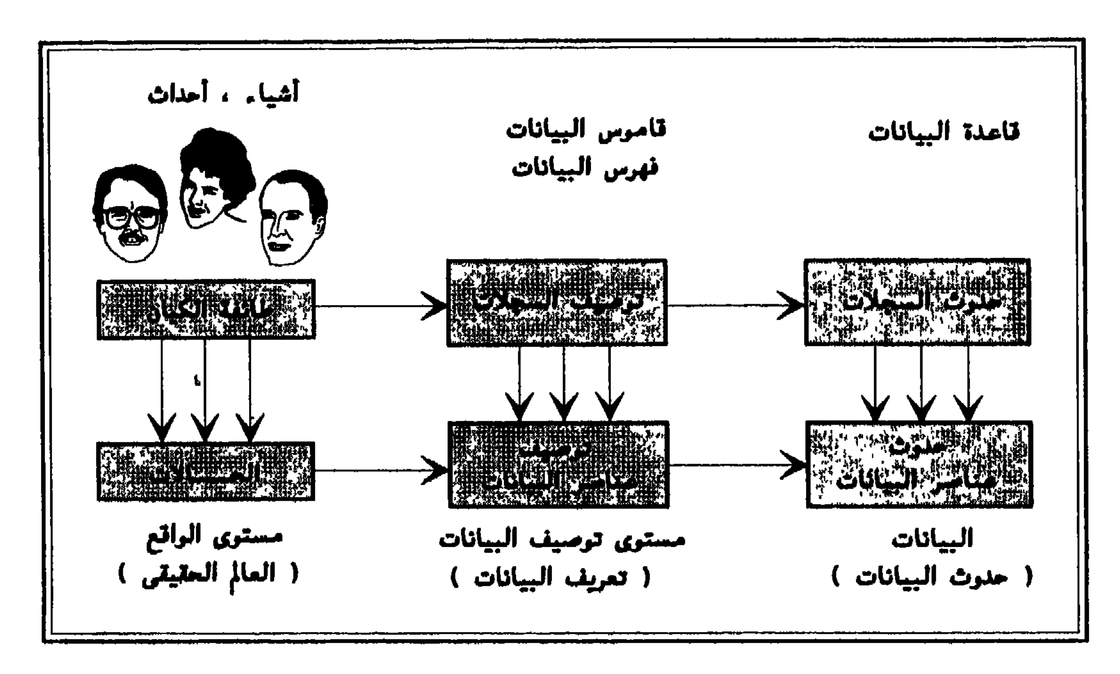


تعرضنا فى الفصل الأول للمفاهيم الأساسية الخاصة بالبيانات. وفى هذا الفصل يتم توضيح خصائص البيانات بشىء من التفصيل وكذلك طرق تمثيل هذه البيانات والربط بينها.

٢ - ١ مستويات تمثيل البيانات

هناك ثلاثة مستويات لتمثيل البيانات تسمى مستويات التجريد (Abstraction Levels) وهى موضحة فى شكل (٢ - ١)

١ - مستوى الواقع (Reality) أو العالم الحقيقى (Real world)
٢ - مستوى توصيف البيانات (Metadata).
٣ - مستوى البيانات الفعلية (Physical Data).



شكل (٢ - ١)

٢ - ١ - ١ مستوى الواقع (Reality)

المقصود بهذا المستوى هو التنظيم أو المنشأة نفسها بما تحتويه من أشخاص

وأشياء أخرى تساهم جميعا فى تحقيق أهداف التنظيم ويشمل المستوى أيضا البيئة المحيطة بالتنظيم (Environment) التى تتأثر به وتؤثر فيه والتى يطلق عليها أيضا كيانات (Entities). وهذه الكيانات قد تكون كيانات مادية محسوسة (Tangible) مثل الموظف أو العميل ، أو كيانات غير محسوسة (Intangible) مثل الحساب (Account) والضريبة (Tax). وهناك فرق بين الكيان نفسه وصنف هذا الكيان (Entity Class) أو نوعه (Entity type). فالمقصود بصنف الكيان أو نوعه هو مجموعة الكيانات التى لها نفس الخصائص مثل العملاء والطلبة والمرضى. بينما الكيان هو أحد عناصر هذه المجموعات مثل عميل محدد أو طالب معين .... إلخ ويسمى أيضا حدوث الكيان (Entity accurence).

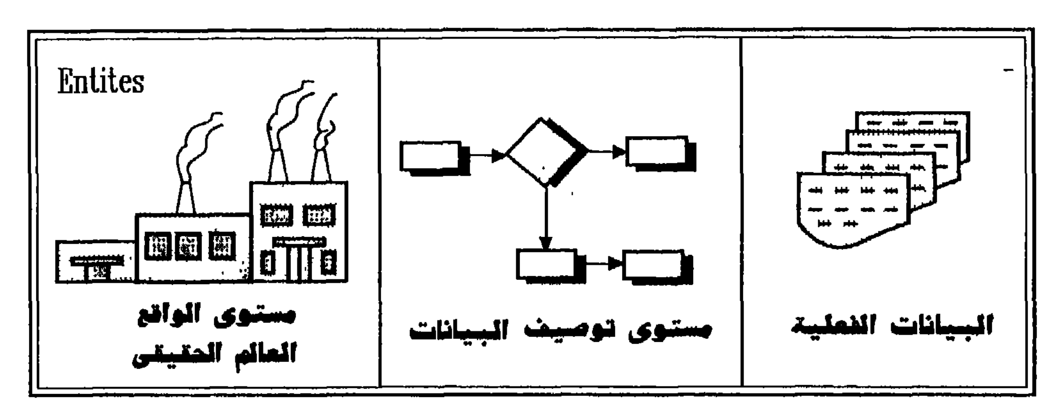
وكل نوع من الكيانات له حقول بيانات خاصة به (Fields) وتسمى أيضا (Attributes) فمثلا إذا كان العملاء (Customers) والمنتجات (Products) هما نوعان من الكيانات فإن الحقول الخاصة بهما يمكن أن تكون كالآتى :

Customers	Products
Customer Number	Product Number
Name	Description
Address	Finish
Telephon Number	Price
Credit Limit	Weight

وكما ذكرنا من قبل أن صنف الكيان (Class) يضم مجموعة الكيانات التى لها نفس الخصائص لذلك يجب أن يكون لكل كيان من هذه المجموعة على الأقل حقل واحد يميزه يسمى المميز (Indentifier) أو حقل المفتاح (Key Field). فمثلا رقم العميل (Customer No.) يمثل المفتاح الخاص بالعميل فى صنف الكيان المسمى بالعملاء (Customers). وفى بعض الحالات يكون المميز أو المفتاح مكونا بالدمج (Concatination) بين حقلين أو أكثر مثل الدمج بين رقم المنتج (Product No.) ورقم الفاتورة (Invoice No.) فى صنف الكيان المسمى بالمنتجات (Products). وهذا ما سيتم إيضاحه فيما بعد عند دراسة الربط (Association) بين الحقول. ويجب ملاحظة أن يكون المميز أو المفتاح منفردا

(Unique) أى لايوجد كيانات لها نفس القيمة الخاصة بهذا الحقل مثل رقم العميل. فلا يمكن أن يكون هناك عميلان لهما نفس الرقم.

والكيانات (Entites) كما سبق الإيضاح هى أجزاء من العالم الحقيقى المتمثل فى التنظيم أو المنشأة والعناصر المحيطة (Environment) والإدارة الناجحة لهذه المنشأة تستلزم إنشاء نماذج توضح خصائص هذه الكيانات ومواصفاتها. وهذه النماذج تغنى عن متابعة وملاحظة هذه الكيانات الحقيقية. ولنا أن نتخيل المدير الذى يكون مضطرا إلى الذهاب إلى كل مخزن ليحصى عدد الأجزاء المخزنة على الرف - وهى كيان حقيقى - عند كل عملية صرف لهذا الصنف. لذلك يلجأ النظام أو المؤسسة إلى إنشاء نماذج توضح خصائص كيان (المخزون) ومواصفاته ، وهذا يقودنا إلى مستوى توصيف البيانات ثم مستوى البيانات الفعلية. أنظر شكل (٢ - ٢).



شكل (٢ - ٢)

٢ - ١ - ٢ مستوى توصيف البيانات (Metadata)

هذا المستوى يتعامل مع مواصفات البيانات بهدف الوصول إلى النموذج المنطقى (Logical Model) لكيانات المنشأة والروابط (Associations) الموجودة بينها. ويستخدم هذه المستوى بواسطة مدير قاعدة البيانات (Database Adminstrator). والذى من خلاله يتم توصيف السجلات والحقول متضمنا أسماء الحقول وأنواعها وأطوالها وإنشاء ما يسمى قاموس البيانات (Data Dictionary) كما يتم توصيف العلاقات والروابط بين الحقول الموجودة فى نوع أو صنف واحد من السجلات وكذلك بين الحقول الموجودة فى أصناف مختلفة.

٢ - ١ - ٣ مستوى البيانات الفعلية (Physical Data)

البيانات الفعلية هى بيانات الكيانات التى تنتمى إلى نوع أو صنف معين وتسمى أيضا حدوث الكيان (Entity Occurance). فإذا كان العميل (Customer) هو أحد أصناف الكيانات فإن (Aly Hasan) يمثل حدوث هذ الكيان ويعتير السجل الخاص به بما يحتويه من حقول هو مستوى البيانات الفعلية التى يتم تمثيلها فى الحاسب. فى حين يكون هناك سجل واحد يتم تعريفه فى مستوى توصيف البيانات (Metadata).

أى أن البيانات الفعلية هى التى يتم تخزينها فى قاعدة البيانات المخزنة فى الحاسب أما توصيف البيانات فإنه لايخزن فى قاعدة البيانات ولكنه يخزن فى قاموس البيانات (Data Dictionary).

مثـــال

لتوضيح الفرق بين المستويات الثلاثة السابق شرحها نفرض أن أحد الأشخاص أراد إنشاء ملف قاعدة بيانات الطلبة (Codets) من خلال برنامج (+ DBase III) لذلك يقوم بكتابة الأمر التالى أمام مشيرة النقطة

```
. CREATE  CADETS
```

وهذا يجعل البرنامج يبدأ فى إنشاء ملف الطلبة ويعرض شاشة إدخال بيانات حتى يقوم المستخدم من خلالها بإدخال بيانات الحقول. أنظر شكل (٢ - ٣).

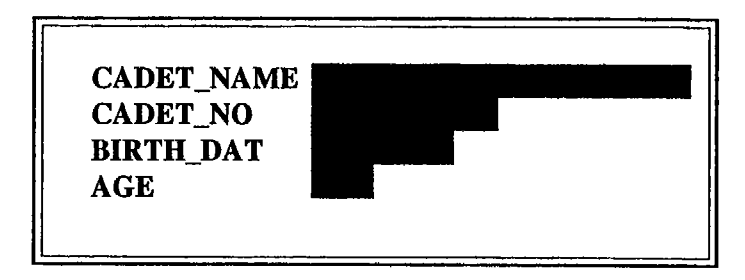
	Field Name	Type	Width	Dec
1	CADET_NAME	Character	30	
2	CADET_NO	Character	10	
3	BIRTH_DAT	Date	8	
4	AGE	Numeric	3	0
5		Character		

شكل (٢ - ٣)

وهذه الشاشة تطلب من المستخدم تعريف كل حقل فى السجل من حيث نوع الحقل (Type) وعرضه (Width) وعدد الكسور العشرية (Dec). هذا التوصيف للسجل هو ما يطلق عليه مستوى توصيف البيانات (Metadata) ، ويلاحظ أن هذا المستوى لايعتمد على بيانات طالب معين. ثم يلى ذلك مستوى إدخال البيانات الفعلية للطلبة (Physical Data) ويتم بكتابة الأمر التالى

. Append

وفى هذه الحالة يعرض البرنامج شاشة إدخال البيانات الموضحة بالشكل (٢ - ٤)



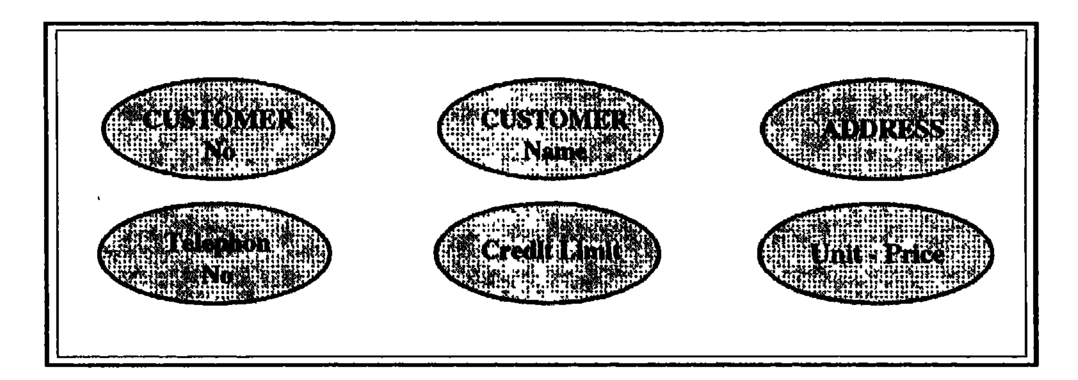
شكل (٢ - ٤)

٢ - ٢ الربط بين البيانات (Data Association)

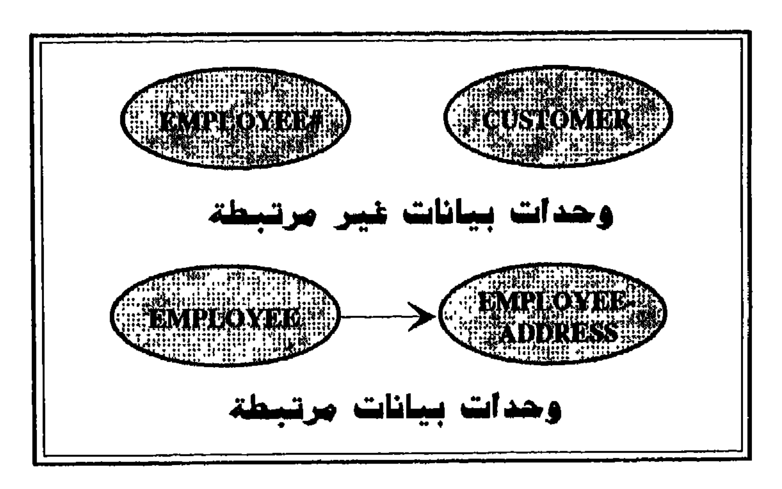
يتم تمثيل الربط بين البيانات عادة بالرسم وفى هذا الجزء يتم توضيح القواعد الأساسية لتمثيل العلاقات بالرسم.

يتم تمثيل وحدة البيانات (Data Item) كما يتضح من شكل (٢ - ٥)

والمقصود بالربط (Association) أن القيم الخاصة بوحدات البيانات المرتبطة تكون معتمدة على بعضها بصورة ما. والشكل (٢ - ٦) يوضح وحدات بيانات مرتبطة وأخرى غير مرتبطة.



شكل (٢ - ٥)



شكل (٢ - ٦)

ومعنى الربط فى هذه الحالة أن كل موظف (Employee) له عنوان (Adress) خاص به ويتم توضيح الربط عن طريق السهم كما يتضح من الشكل.

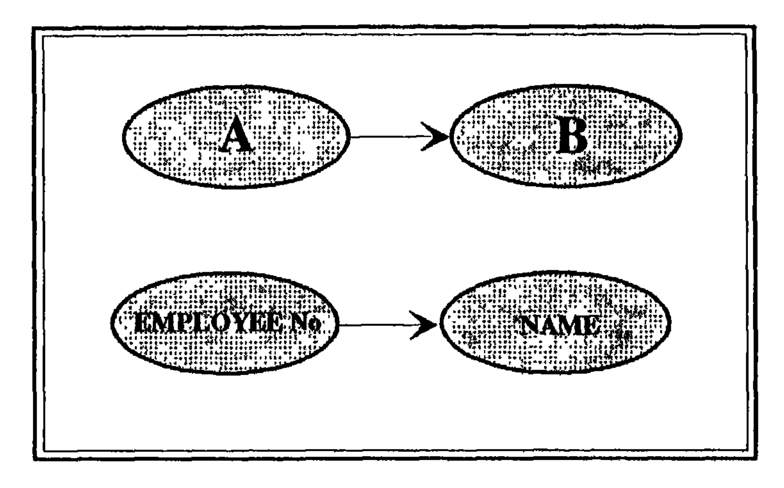
٢ - ٣ أنــواع الربــط

هناك ثلاثة أنواع من الربط :

١ - الربط الأحادى (One Association)
٢ - الربط المتعدد (Many Association)
٣ - الربط المشروط (Conditional Association)

٢ - ٣ - ١ الربط الأحادى (One Association)

الربط الأحادى من وحدة البيانات (A) إلى وحدة البيانات (B) ، كما يتضح من الشكل (٢ - ٧) يعنى أن كل قيمة لوحدة البيانات (A) يقابلها قيمة واحدة فقط لوحدة البيانات (B) تكون مرتبطة بها ويتم تمثيل هذه العلاقة بسهم ذى رأس واحد فمثلا كل موظف له عنوان محدد.

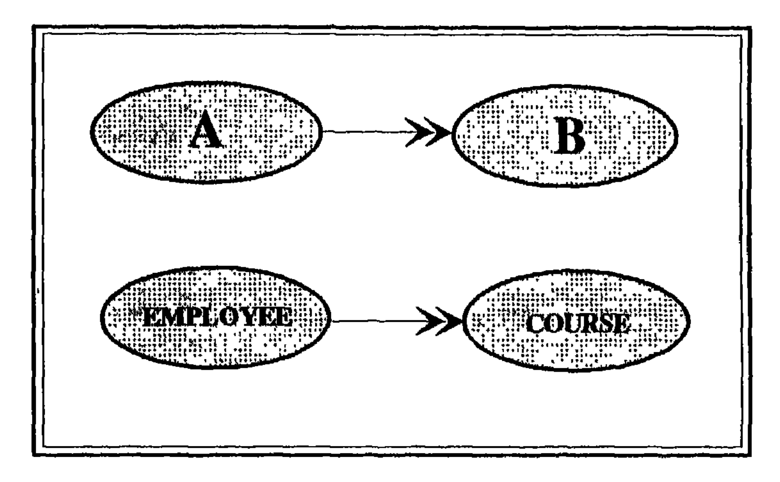


شكل (٢ - ٧)

٢ - ٣ - ٢ الربط المتعدد (Many Association)

الربط المتعدد من وحدة البيانات (A) إلى وحدة البيانات (B) يعنى أن كل قيمة لوحدة البيانات (A) يقابلها قيمة واحدة أو عدة قيم أو لايقابلها أى قيمة لوحدة البيانات (B) ويتم تمثيل هذه العلاقة بسهم ذى رأسين كما يتضح من شكل (٢ - ٨).

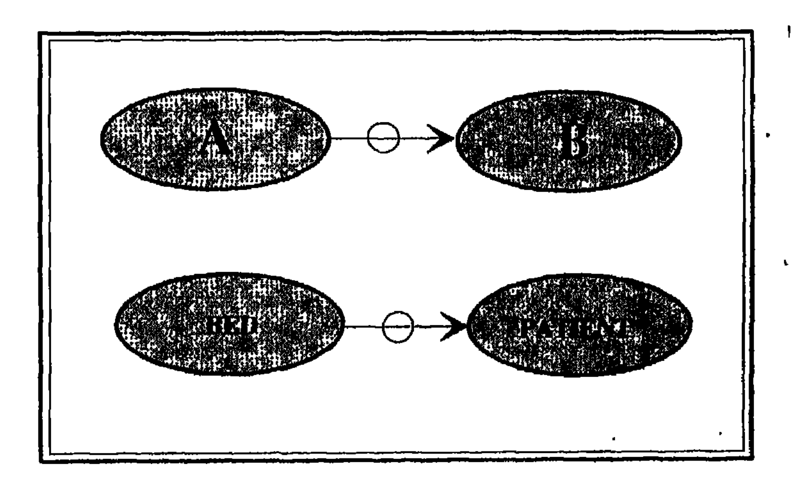
فمثلا كل موظف قد يكون حصل على حلقات تدريبية (Courses) أو حصل على حلقة واحدة أو لم يحصل على أى حلقة.



شكل (٢ - ٨)

٢ - ٣ - ٣ الربط المشروط (Conditional Association)

الربط المشروط من وحدة البيانات (A) إلى وحدة البيانات (B) يعنى أن كل قيمة لوحدة البيانات (A) قد يقابلها قيمة واحدة لوحدة البيانات (B) وذلك بناء على شرط معين ويتم تمثيل ذلك بسهم وعليه دائرة كما يتضح من الشكل (٢ - ٩). فمثلا فى قاعدة بيانات خاصة بمستشفى نجد أن كل سرير فى المستشفى قد يخصص لمريض معين وقد لايكون مخصصا لأى مريض فى وقت محدد.



شكل (٢ - ٩)

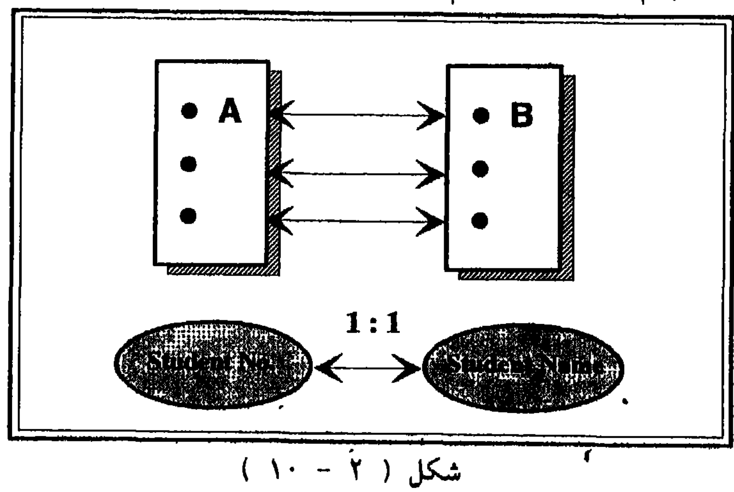
٢ - ٤ الربـــط المتبـــادل (Mutual Associaltion)

الربط المتبادل هو الربط فى إتجاهين أى أن هناك ربطا من وحدة البيانات (A) إلى وحدة البيانات (B) والعكس. وهناك ثلاثة أنواع من الربط المتبادل.

١ - الربط من واحد إلى واحد (One - to - One).
٢ - الربط من واحد إلى كثيرين (One - to - Many).
٣ - الربط من كثيرين إلى كثيرين (Many - to - Many).

٢ - ٤ - ١ الربط المتبادل من واحد إلى واحد (One - to - One Association, 1 : 1)

ويعنى هذا الربط أن كل قيمة لوحدة البيانات (A) يقابلها قيمة واحدة فقط لوحدة البيانات (B) مرتبطة بها وكذلك كل قيمة لوحدة البيانات (B) يقابلها قيمة لوحدة البيانات (A) مرتبطة بها. فمثلا كل رقم طالب يقابله إسم طالب معين ، وكذلك أى إسم طالب يقابله رقم طالب معين. أنظر شكل (٢ - ١٠).

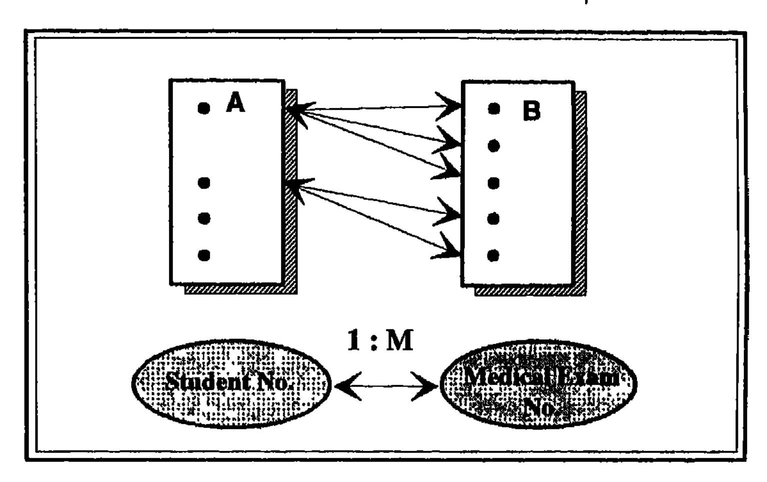


شكل (٢ - ١٠)

٢ - ٤ - ٢ الربط المتبادل من واحد إلى كثيرين (One - to - Many Association, 1 : M)

فى هذا النوع من الربط نجد أن كل قيمة لوحدة البيانات (A) يقابلها قيمة أو أكثر أو لايقابلها قيمة لوحدة البيانات (B) وكل قيمة لوحدة البيانات (B) يقابلها

قيمة واحدة لوحدة البيانات (A) مثل العلاقة بين رقم الطالب ورقم الفحص الطبى حيث أن كل طالب قد يؤدى فحصا طبيا واحدا أو عدة فحوص أولا يؤدى أى فحص فى حين يختص كل رقم فحص بطالب محدد واحد. أنظر شكل (٢ - ١١)



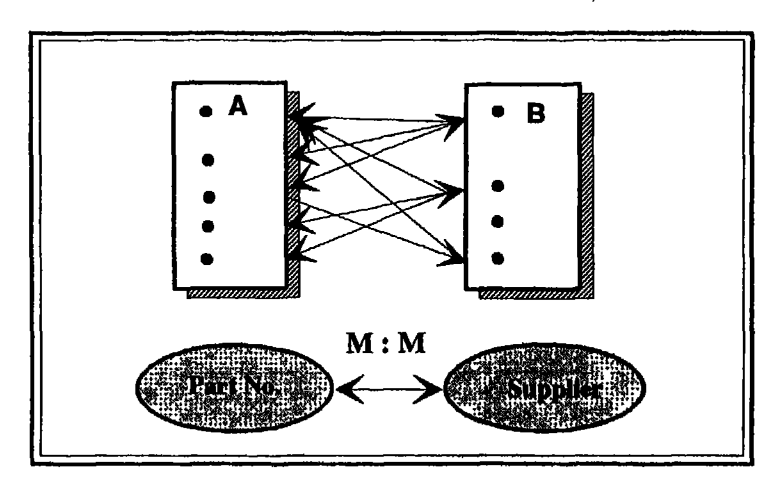
شكل (٢ - ١١)

٢ - ٤ - ٣ الربط المتبادل من كثيرين إلى كثيرين (Many - to - Many Association, M : N)

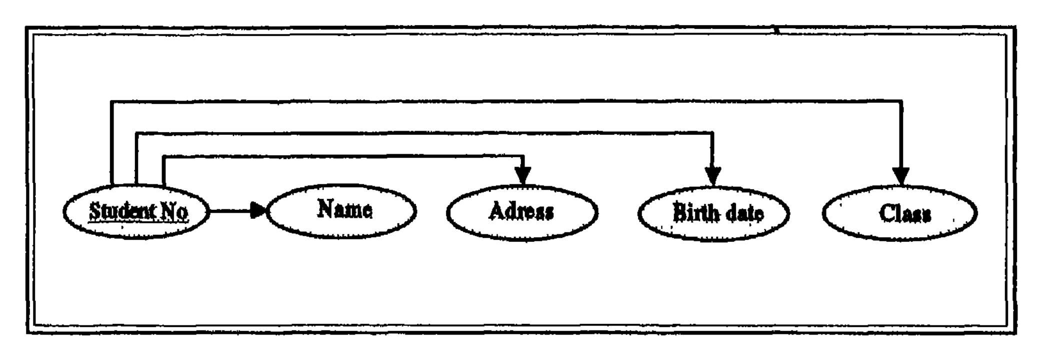
يعنى هذا الربط أن كل قيمة لوحدة البيانات (A) يقابلها قيمة أو أكثر أو لا يقابلها قيمة لوحدة البيانات (B). كما أن كل قيمة لوحدة البيانات (B) يقابلها قيمة أو أكثر أولا يقابلها قيمة لوحدة البيانات (A). فمثلا العلاقة بين رقم القطعة والمورد تمثل الربط من كثيرين إلى كثيرين. فمثلا كل قطعة برقم معين (Part No.) قد يتم توريدها من عدة موردين ، كما أن المورد الواحد يمكن أن يورد أكثر من قطعة. أنظر شكل (٢ - ١٢).

٢ - ٥ الربـط بيـن الحقـــول

كما سبق الإيضاح فإن السجل يتكون من عدة حقول لذلك فمن الطبيعى أن يكون هناك ربط بين الحقول داخل كل سجل. والحقل الفهرسى (Key Field) والذى يستخدم فى تمييز كل سجل ، يرتبط بكل حقول السجل كما يتضح من الشكل (٢ - ١٣) ويلاحظ وجود خط تحت الحقل الفهرسى لتمييزه.

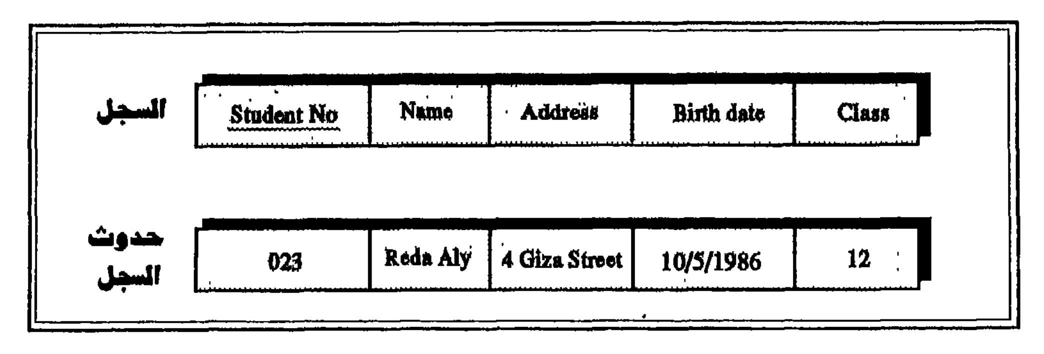


شكل (٢ - ١٢)



شكل (٢ - ١٣)

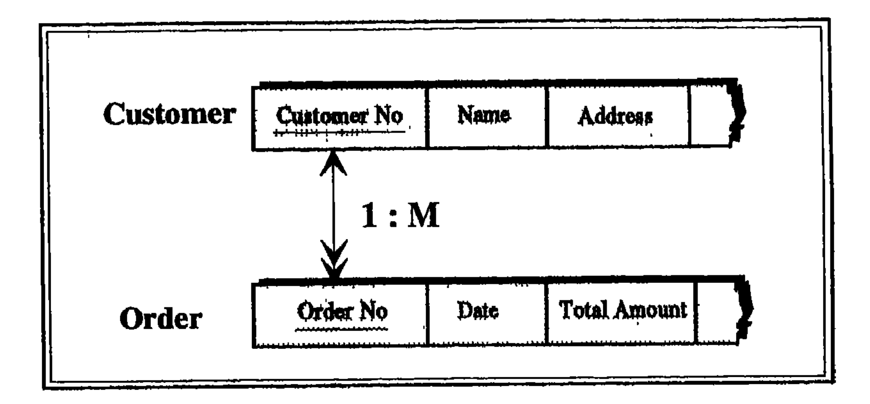
كما يمكن تمثيل السجل بمستطيلات كالموضحة بالشكل (٢ - ١٤) والذى يوضح أيضا الحدوث (Occurance) لمثل هذا السجل.



شكل (٢ - ١٤)

٢ - ٦ الربط بين السجلات

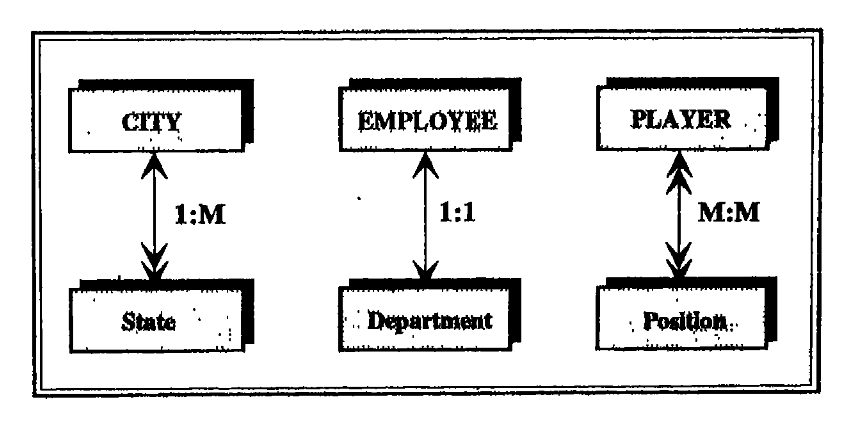
هناك أيضا ربط بين السجلات ويكون عادة ربطا متبادلا. فمثلا الشكل (٢ - ١٥) يوضح سجلين مرتبطين عن طريق حقل المفتاح فى كل منها.



شكل (٢ - ١٥)

حيث أن كل عميل يقابله طلب شراء (Order) أو عدة طلبات أى أن العلاقة فى هذه الحالة هى (1 : M).

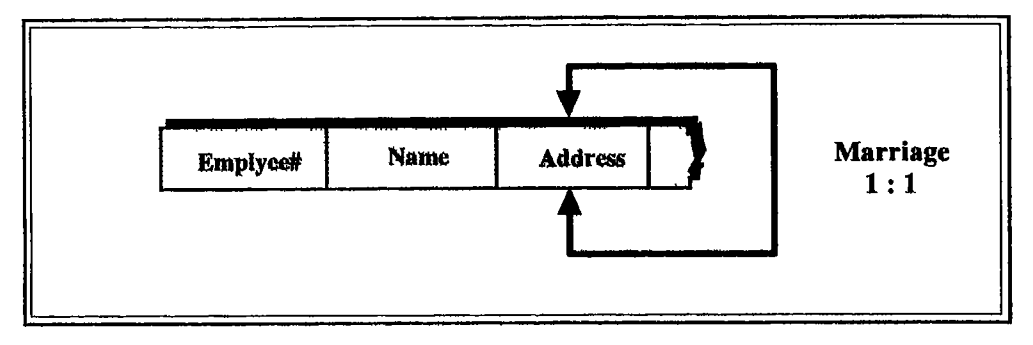
والشكل (٢ - ١٦) يوضح نماذج مختلفة للعلاقة بين السجلات.



شكل (٢ - ١٦)

٢ - ٧ الربــط الذاتـى (Recursive Association)

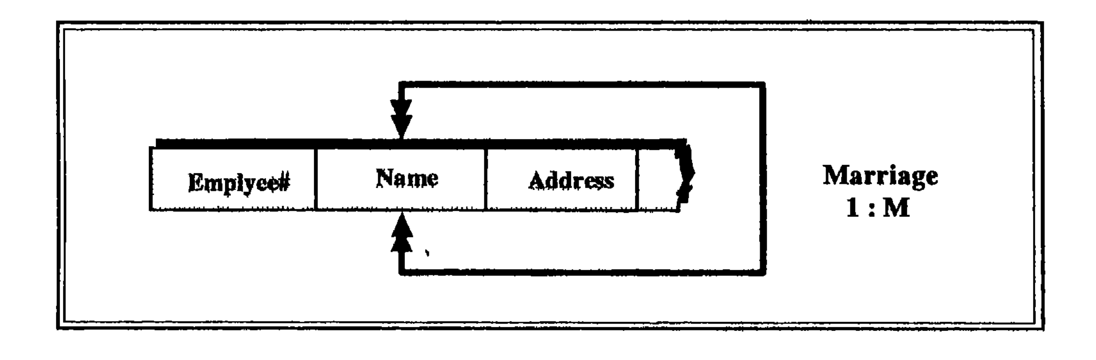
فى بعض الأحيان تقوم السجلات بتوصيف نوعين من الكيانات (Entities) مثل سجل الموظفين والذى يحتوى على موظفين ذكور وإناث قد يكون بعضهم متزوجا من بعض يكون هناك ربط ذاتى داخل الكيان الذى يقوم السجل بتوصيفه. أنظر شكل (٢ - ١٧).



شكل (٢ - ١٧)

ومن هذا الشكل يتضح أن هناك ربطا ذاتيا بين الموظف ونفسه عن طريق الزواج (Marriage) ونوع هذا الربط من واحد الى واحد (1 : 1).

والشكل (٢ - ١٨) يوضح نوعا آخر من الربط الذاتى من واحد الى كثيرين (1 : M).



شكل (٢ - ١٨)

حيث أن الموظف قد يكون مديرا لعدد من الموظفين.

الفصل الثالث

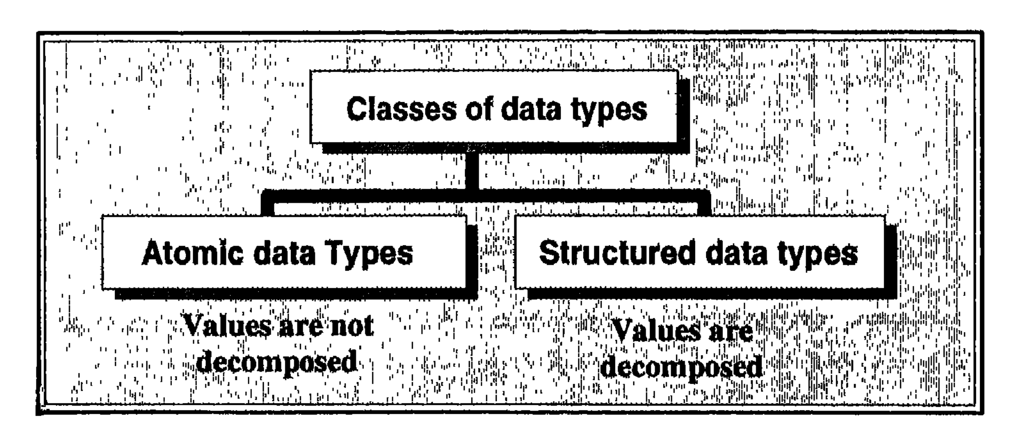
تراكيب البيانات

(Data Structures)



يمكن تعريف مصطلح نوع البيانات (Data Type) على أنه مجموعة القيم التى تعطى لمجموعة من الأشخاص أو الأشياء حتى يمكن تمييزها كقيم معرفة بالإضافة إلى مجموعة العمليات التى يمكن إجراؤها على هذه القيم.

ويمكن تقسيم نوع البيانات إلى نوعين رئيسين ، النوع الأول هو نوع البيانات الأولى (Atomic Data Type) مثل الأرقام الصحيحة (Integer No.) ويمكن إجراء العمليات الحسابية عليها مثل الجمع والطرح والضرب والقسمة ... إلخ). والنوع الثانى هو نوع البيانات المركب (Structrured Data Type) وهى البيانات التى يمكن تحليلها إلى مجموعة من الوحدات بعضها عنصرى والبعض الآخر مركب كذلك فإن هذه البيانات تربطها ببعضها علاقات (Relationships) معينة أنظر شكل (٣ - ١).



شكل (٣ - ١)

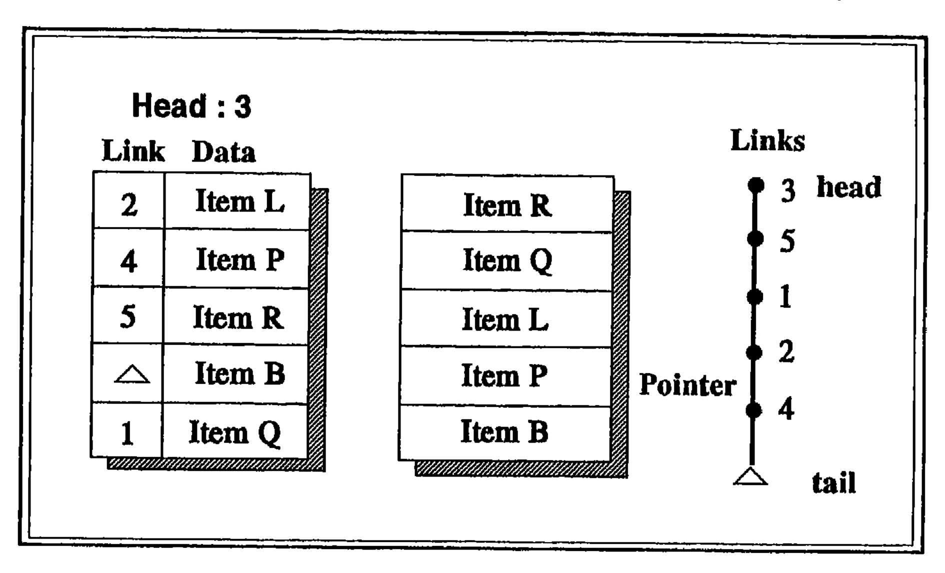
وفى هذا الفصل سوف نناقش أنواع تراكيب البيانات التى لها علاقة مباشرة بتشغيل قاعدة البيانات. ومن هذه التراكيب القوائم المرتبطة (Linked Lists) والتى تستخدم لتمثيل العلاقات الفعلية (Physical Relationships) للبيانات. وكذلك الشجرة (Tree) والشبكة (Network) التى تستخدم فى تمثيل العلاقات المنطقية (Logical Relationships).

٣ - ١ القوائم المرتبطة (Linked Lists)

القائمة المرتبطة هى مجموعة من وحدات البيانات المرتبة (Ordered) وكلمة مرتبطة (Linked) تعنى أن عملية الترتيب تتم باستخدام روابط (Links) أو مؤشرات

(Pointers) موجودة داخل وحدات البيانات. وتتميز القائمة المرتبطة بوجود مؤشر لأول وحدة بيانات يسمى رأس القائمة (Head) وكذلك مؤشر فى آخر وحدة بيانات فى القائمة يسمى ذيل القائمة (Tail).

والشكل (٣ - ٢) يوضح مثالا لقائمة مرتبطة (Linked List) موجودة فى مصفوفة (Array) ومثيلتها الفعلية.



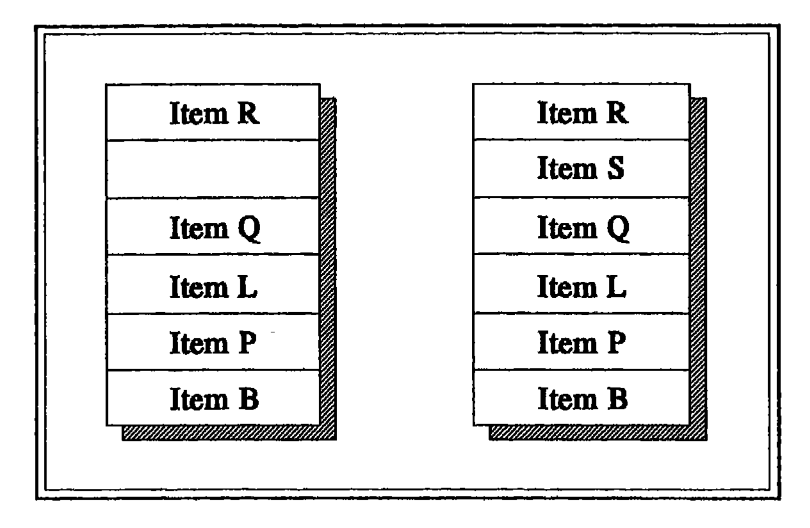
شكل (٣ - ٢)

ويلاحظ أن كل صف من القائمة المرتبطة يمثل وحدة بيانات. الرقم الأول فى كل صف فى القائمة يمثل الربط (Link) وهو رقم الصف الذى يتوجه إليه المؤشر لقراءة وحدة البيانات منه فمثلا رأس القائمة (Head) له القيمة (3) والتى تعنى أن أول وحدة بيانات فى المصفوفة تقع فى الصف الثالث فى القائمة. أى أن وحدة البيانات (R) تقع فى الصف الأول فى المصفوفة. كذلك وحدة البيانات (R) لها رقم ربط يشير إلى الصف الخامس أى أن وحدة البيانات (Q) تقع فى الصف الثانى فى المصفوفة. وهكذا فإن ترتيب المصفوفة كلها هو (R - Q - L - P - B).

لاحظ أن ترتيب القائمة المرتبطة ليس مثل الترتيب الفعلى فى المصفوفة.

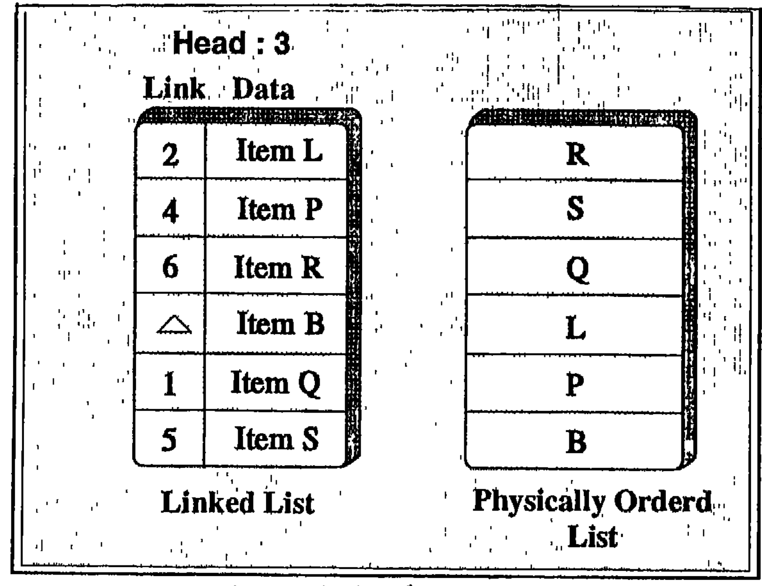
وتتميز القوائم المرتبطة عن القوائم الفعلية فى سهولة عمليات التشغيل. فمثلا لحشر وحدة بيانات (S) بين وحدة البيانات (R) ، (Q) فى قائمة الترتيب الفعلى فإنه يلزم

تحريك بقية الوحدات (Q) ، (L) ... (B) لأسفل صفا واحدا كما يتضح من الشكل (٣ - ٣).



شكل (٣ - ٣)

أما فى حالة القائمة المرتبطة فإننا ببساطة نضيف الوحدة (S) فى نهاية القائمة ونغير مؤشرين فقط. فرقم الرابط أمام الوحدة (R) يتم تغييره ليشير إلى الوحدة (S) ثم نضيف مؤشرا للوحدة (Q) فى الوحدة (S) كما يتضح من الشكل (٣ - ٤).

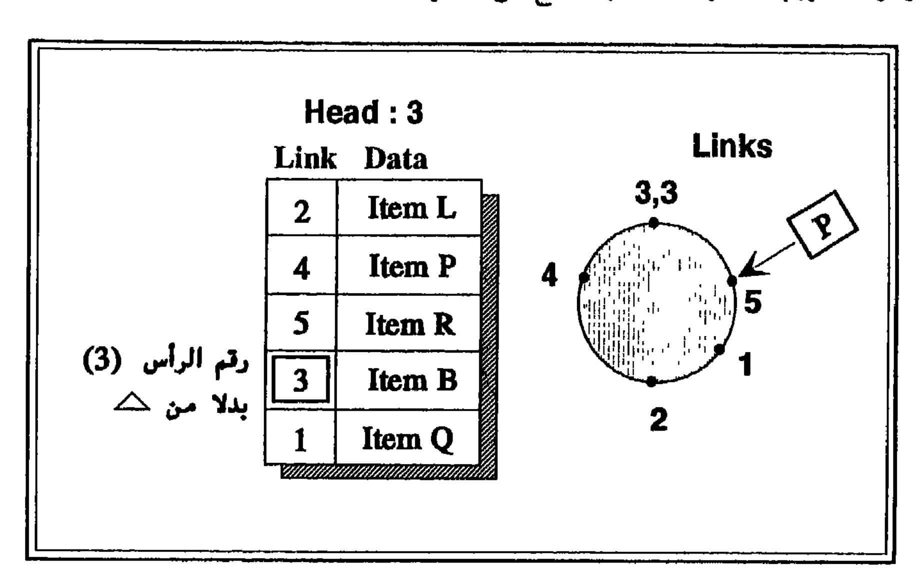


شكل (٣ - ٤)

كذلك فى حالة حذف وحدة بيانات فإنه يلزم لذلك تغيير مؤشر (Pointer) واحد. وهذه المميزات للقوائم المرتبطة تجعلها أداة فعالة لتمثيل تراكيب البيانات المعقدة كما سيتضح لنا فيما بعد.

٣ - ١ - ١ القوائم المرتبطة الدائرية والثنائية (Circular and Two - Way Linked Lists)

القائمة المرتبطة الدائرية تعنى قائمة مرتبطة يحذف منها المؤشر الخاص (Δ) والذى يشير إلى نهاية القائمة ونضع بدلا منه مؤشر لرأس القائمة بحيث تكون مؤشرات الروابط دائرة كما هو واضح فى شكل (٣ - ٥).

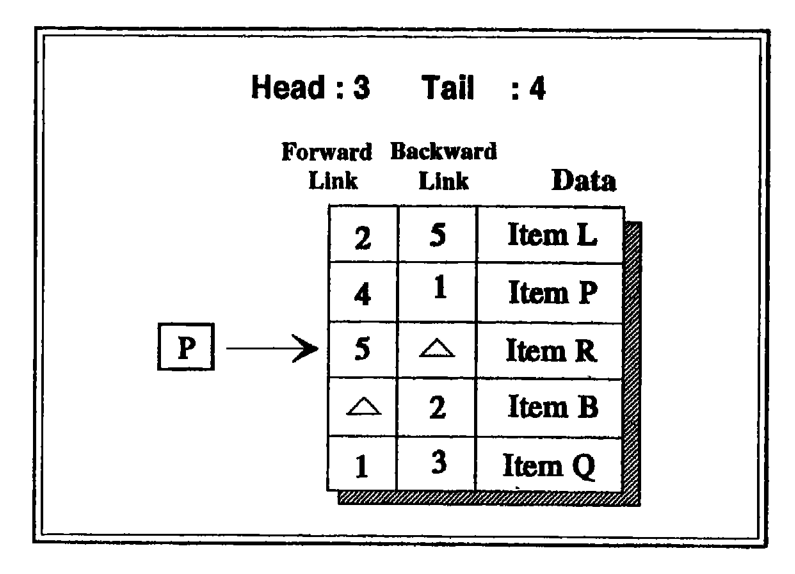


شكل (٣ - ٥)

لاحظ أن (Δ) حل محلها رقم (3) وهو رابط لرأس القائمة. وتتميز قائمة الربط الدائرية بأنه يمكن الوصول إلى أى وحدة بيانات من أي وحدة بيانات أخرى وليس بالضرورة من رأس القائمة.

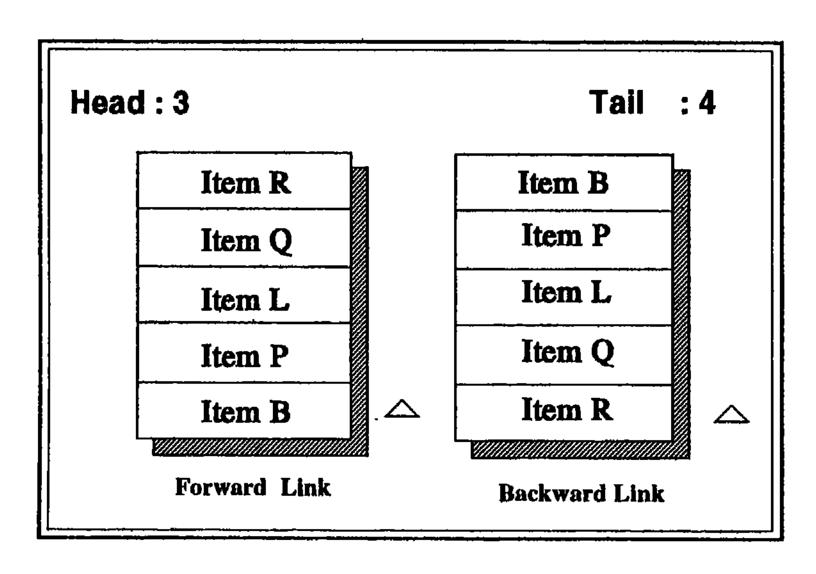
أما القائمة المرتبطة الثنائية (Two - Way Linked List) فهى تسمح بتشغيل البيانات فى إتجاهين أمامى (Forward) أو خلفى (Backward) وهى تحتاج إلى حقلين من الروابط لكل وحدة بيانات. أحدهما للإرتباط فى الإتجاه الأمامى والآخر

للإرتباط فى الإتجاه الخلفى كما يتضح من الشكل (٣ - ٦)



شكل (٣ - ٦)

يلاحظ وجود مؤشر ذيل (Tail) وهو رابط لآخر وحدة بيانات فى القائمة يستخدم فى حالة الإرتباط فى الإتجاه الخلفى. والشكل (٣ - ٧) يوضح الترتيب الطبيعى لوحدات البيانات بإستخدام الربط الأمامى والخلفى.

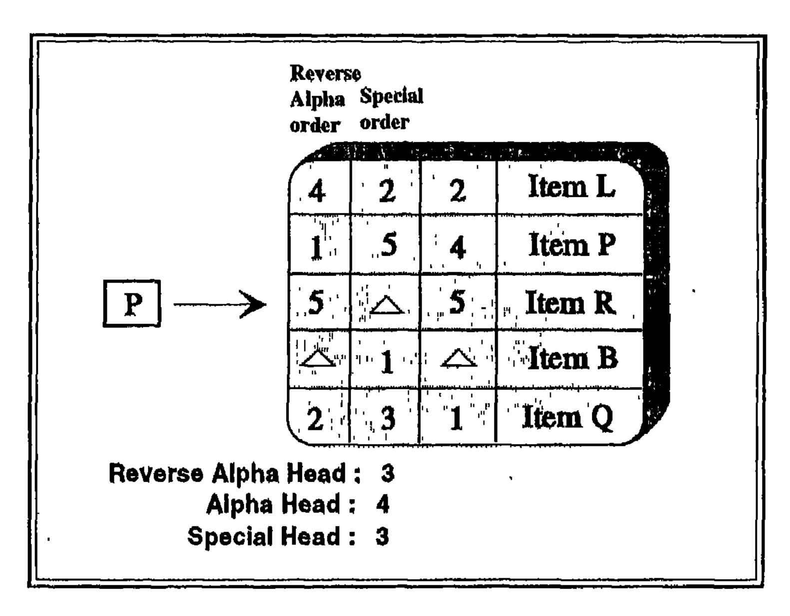


شكل (٣ - ٧)

ويمكن تحويل القائمة المرتبطة الثنائية إلى قائمة مرتبطة ثنائية دائرية وذلك بحذف علامة (Δ) واستبدالها برقم رأس القائمة (Head) (فى حالة الإرتباط الأمامى) أو رقم ذيل القائمة (Tail) (فى حالة الإرتباط الخلفى).

٣ - ١ - ٢ القوائم متعددة الإرتباط (Multiple Linked Lists)

ويستخدم فى هذه القوائم أكثر من حقلين للروابط مع مجموعة من وحدات البيانات. ويفضل إستعمال هذه القوائم فى حالة تشغيل وحدات البيانات بإستمرار وبترتيب مختلف. والشكل (٣ - ٨) يوضح تمثيل إحدى القوائم متعددة الإرتباط.



Reverse Alpha order		Special order	
4	2	2	Item L
1	5	4	Item P
5	Δ	5	Item R
Δ	1	Δ	Item B
2	3	1	Item Q

شكل (٣ - ٨)

وتستخدم القوائم المرتبطة بأنواعها المختلفة لإدارة السجلات فى أنظمة الملفات المباشرة.

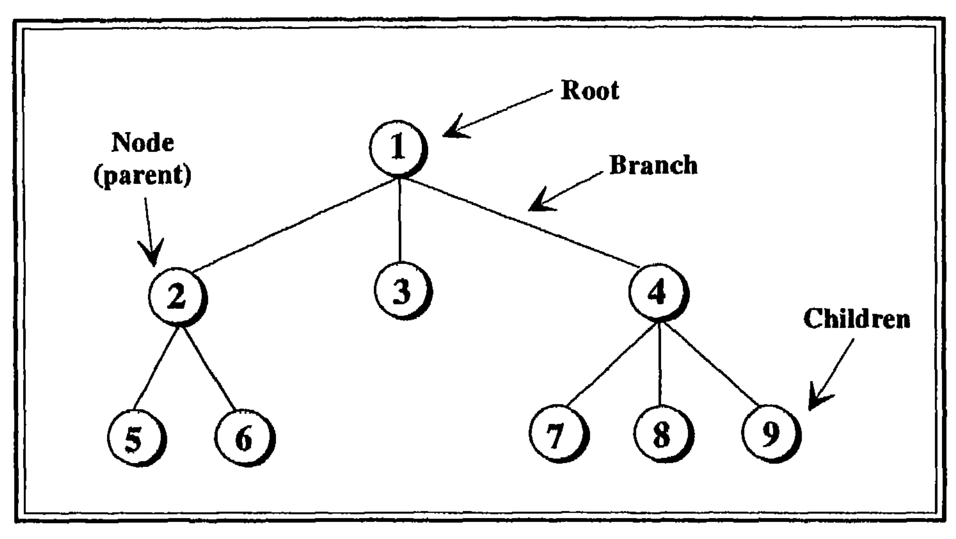
٣ - ٢ الشجـــــرة (Tree)

من أساليب تراكيب البيانات التى تستخدم فى محيط تشغيل قواعد البيانات أسلوب الشجرة (Tree) والذى يستخدم لتمثيل التراكيب المنطقية (Logical) للبيانات وفى هذا

الجزء سوف نتعرض بالتحليل لأهم سمات أسلوب الشجرة والأنماط المختلفة لتمثيلها.

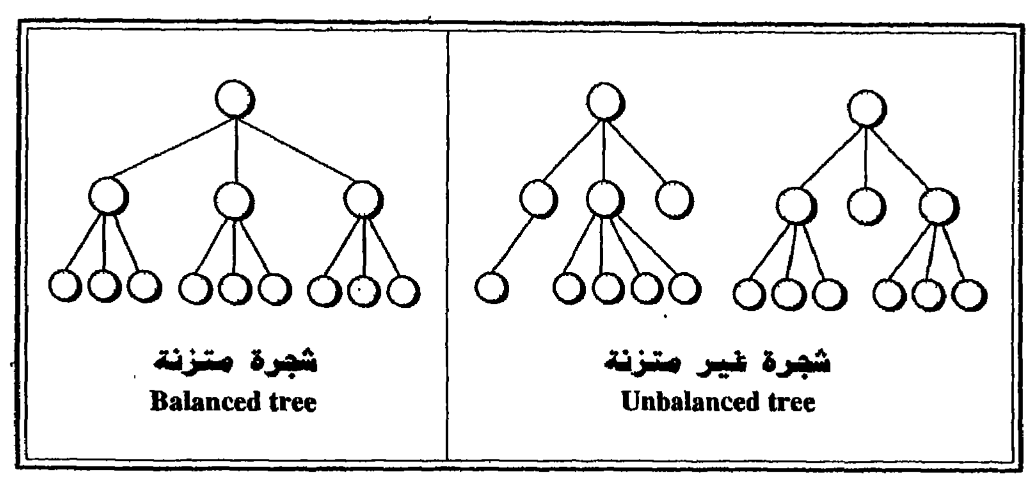
٣ - ٢ - ١ تعريف

الشجرة كما فى شكل (٣ - ٩) عبارة عن نظام متفرع تدريجيا يتكون من مجموعة من العقد أو نقاط الربط (Nodes) ، وتمثل نقطة الربط بدائرة وتتصل مع بعضها بأفرع (Branches). وتسمى نقطة الربط فى أعلى الشجرة الجذر(Root). وكل نقطة ربط ، فيما عدا الجذر ، لها أب (Parent) وهو نقطة الربط التى تسبقها والمتصلة معها بفرع (Branch). وكل مجموعة نقاط لها أب معين تسمى أبناء (Childern) وليس هناك حد أقصى لعدد الأبناء.



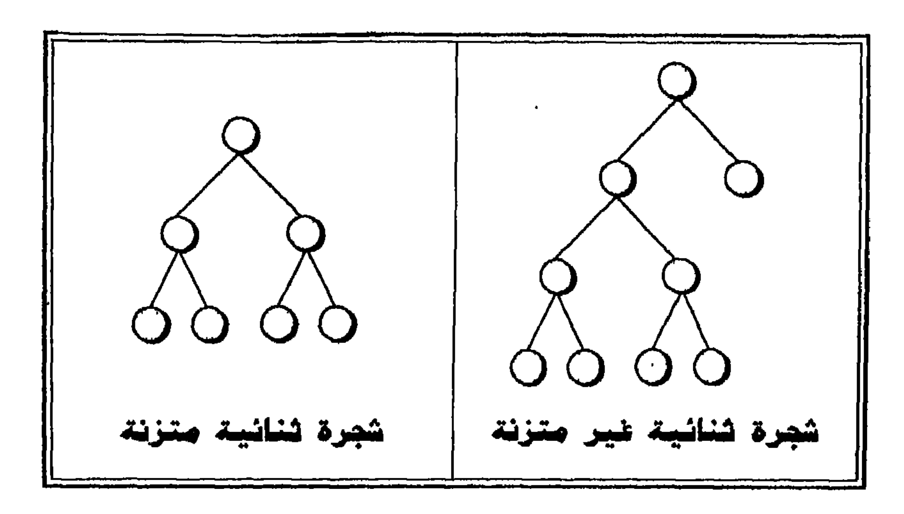
شكل (٣ - ٩)

فمثلا نقطة الربط (2) هى الأب للنقطة (5) والنقطة (4) أب للنقطة (8) كذلك تسمى النقط (6 , 5) وأيضا (9 , 8 , 7) أبناء. وتسمى النقاط التى لها نفس الأب بالتوائم (Twins). وتسمى الشجرة التى يتساوى فيها عدد القوائم لكل أب بالشجرة المتزنة (Balanced Tree). وإذا لم يتساوى عدد القوائم لكل أب تسمى شجرة غير متزنة (Unbalaced Tree) والشكل (٣ - ١٠) يوضح أنواعا مختلفة من الأشجار (Trees).



شكل (٣ - ١٠)

والشجرة الثنائية (Binary Tree) لايزيد عدد الأبناء فيها عن إثنين ويمكن أن تكون متزنة أو غير متزنة كما يتضح من الشكل (٣ - ١١).

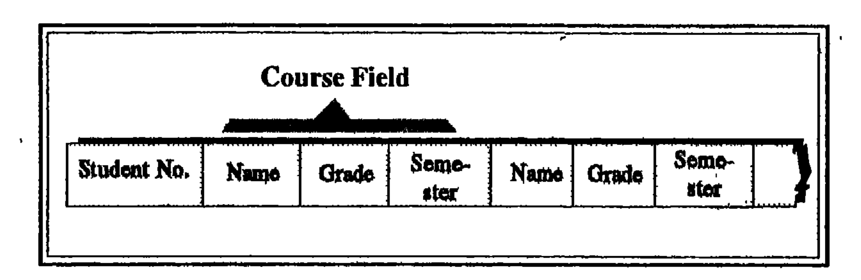


شكل (٣ - ١١)

ويمكن تحويل أى نوع من الأشجار إلى شجرة ثنائية مما يسهل التمثيل الطبيعى لوحدات البيانات.

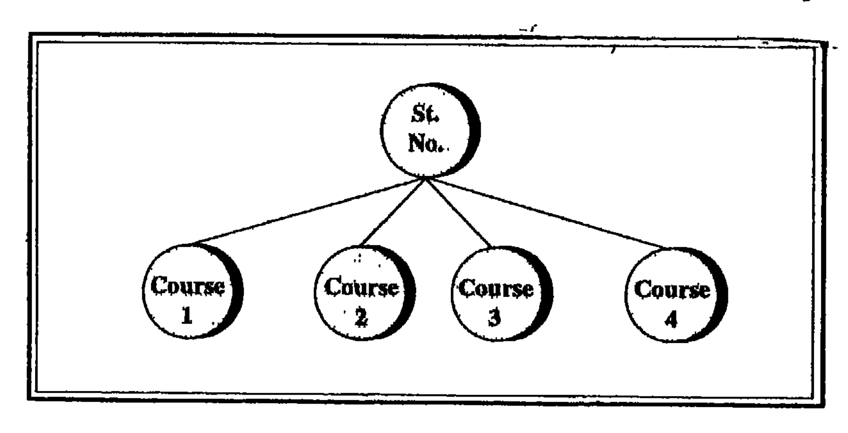
٣ - ٢ - ٢ تطبيقات على نظم قواعد البيانات

كما سبق الإيضاح فإن نظام الأشجار (Trees) غالبا ما يستخدم لتمثيل علاقات البيانات المنطقية ولعل أبسط الأمثلة هو مثال تكرار حقل معين فى سجل قاعدة بيانات. والشكل (٣ - ١٢) يوضح سجل قاعدة بيانات خاصة بالطلبة



شكل (٣ - ١٢)

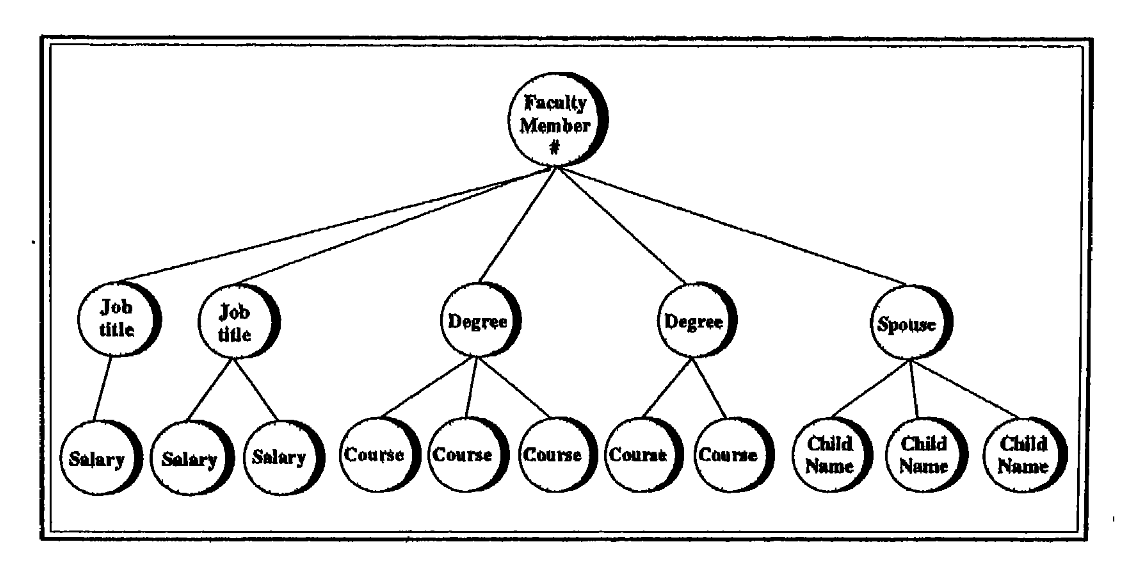
وحيث أن الطالب عليه أن يتم عددا معينا من المواد (Courses) لذلك فإن سجل قاعدة البيانات سوف يكون له أطوال (Lengths) مختلفة فمثلا السجل الخاص بطالب السنة النهائية أطول من السجل الخاص بطالب فى السنة الأولى والتركيب المنطقى للسجل عند حدوث البيانات (Occurrence) يمكن تمثيله بالشجرة فى شكل (٣ - ١٣).



شكل (٣ - ١٣)

وهنا الجذر (Root) هو رقم الطالب والأبناء هم المواد (إسم المادة - الدرجة - الفصل الدراسى).

مثال آخر سجل قاعدة بيانات خاصة بعضو هيئة تدريس فى جامعة ما وهذا السجل يحتوى على رقم وإسم العضو كذلك تاريخه الوظيفى (Job Title) وهو حقل متكرر ودرجاته العلمية (Degree) وهو حقل متكرر والمواد التى درسها (Courses) للحصول على كل درجة وأخيرا بياناته الشخصية مثل الحالة العائلية (Spouse) وعدد الأطفال. ولأن علاقات البيانات (Data Relationships) يصعب وصفها بالكلمات أو بمخطط السجل لذلك فمن السهل تمثيلها بشجرة. ويمثل شكل (٣ - ١٤) حدوث هذه الشجرة.



شكل (٣ - ١٤)

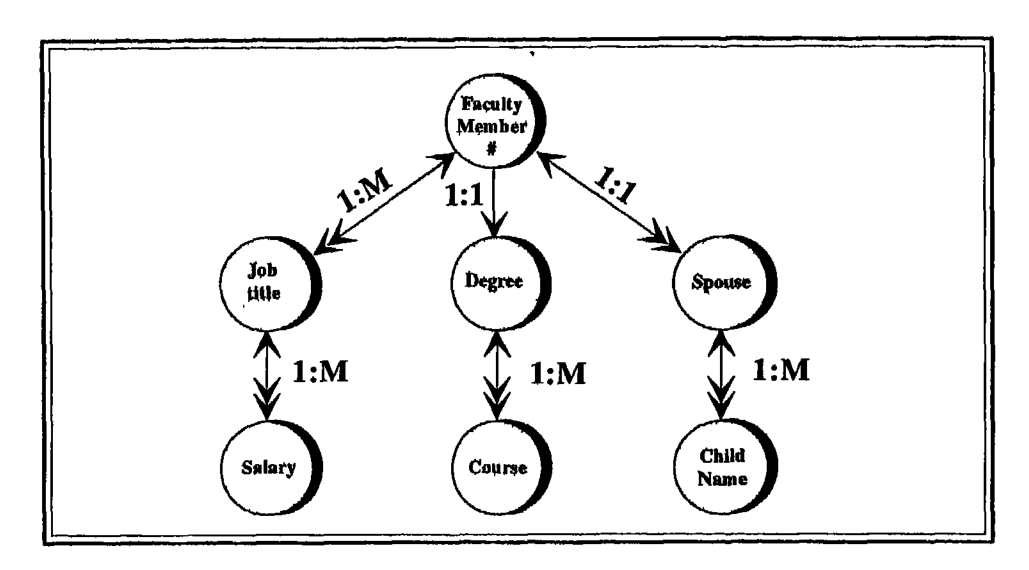
ويمكن استخدام تمثيل الربط المتبادل مع طريقة الشجرة للتعبير عن المثال السابق كما يتضح من الشكل (٣ - ١٥).

٣ - ٣ الشبـــكة (Network)

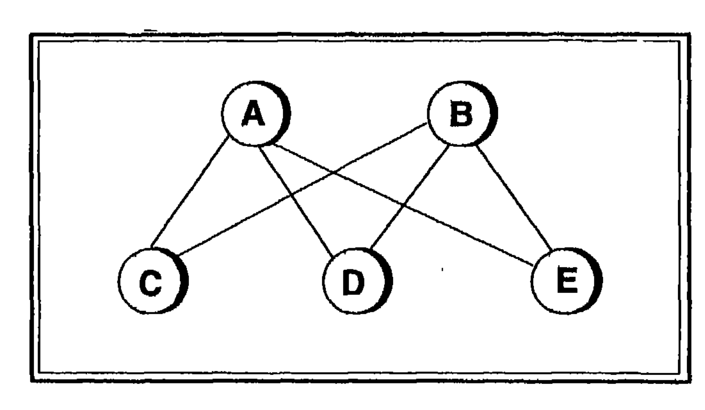
٣ - ٣ - ١ تعريفـــات

تتكون الشبكة من مجموعة من نقاط الربط أو العقد (Nodes) والأفرع (Branches) مثل الشجرة ولكنها تختلف عنها فى أن الإبن (Children) الواحد يمكن أن يكون له أكثر من أب (Parent). أى أن العلاقة بين الأبن والأب أو الأب

والإبن يمكن أن تكون واحد إلى كثيرين (1 : M) والشكل (٣ - ١٦) يبين الحدوث (Occurrence) الخاص بشبكة معينة.



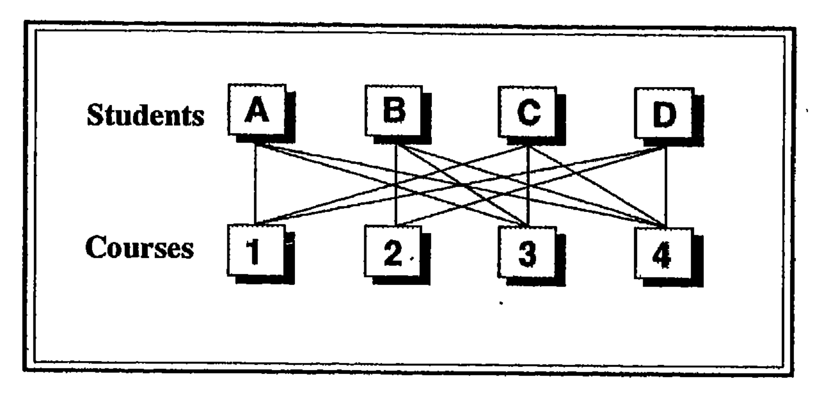
شكل (٣ - ١٥)



شكل (٣ - ١٦)

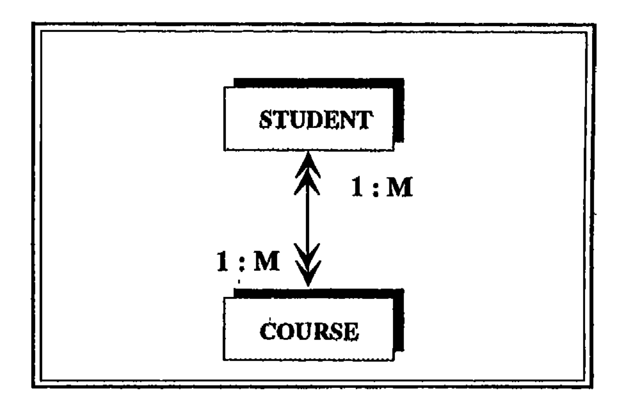
ويلاحظ أن لكل من الأبناء (C , D , E) الآباء (A , B).

وعلى سبيل المثال عند التسجيل(Registration) لعدد من الطلبة (Students) لمواد دراسية معينة فإن العلاقة بين كل طالب والمواد علاقة (1 : M) والعلاقة بين المادة والطلبة (1 : M) فالطالب (A) يمكن أن يسجل للمواد (1, 3 , 4) والمادة (1) مسجل لها الطلبة (A , C , D) كما يتضح من الشكل (٣ - ١٧)



شكل (٣ - ١٧)

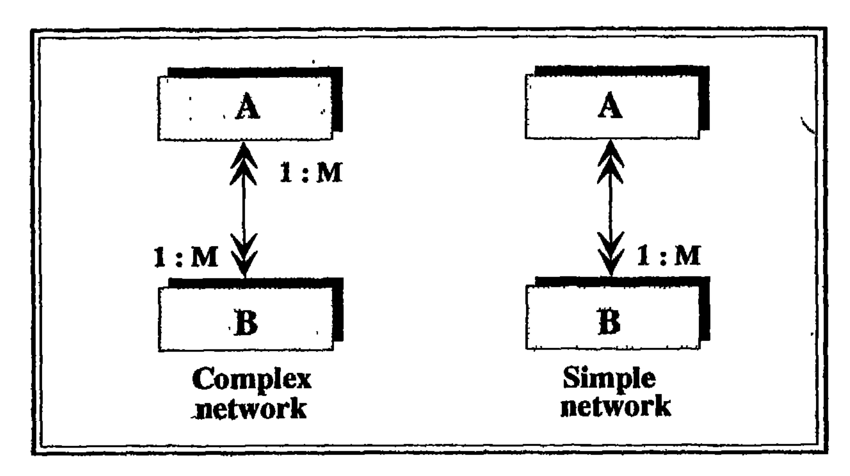
ونفس المثال يمكن توضيحه باستخدام طريقة الربط المتبادل ، كما فى الشكل (٣ - ١٨)



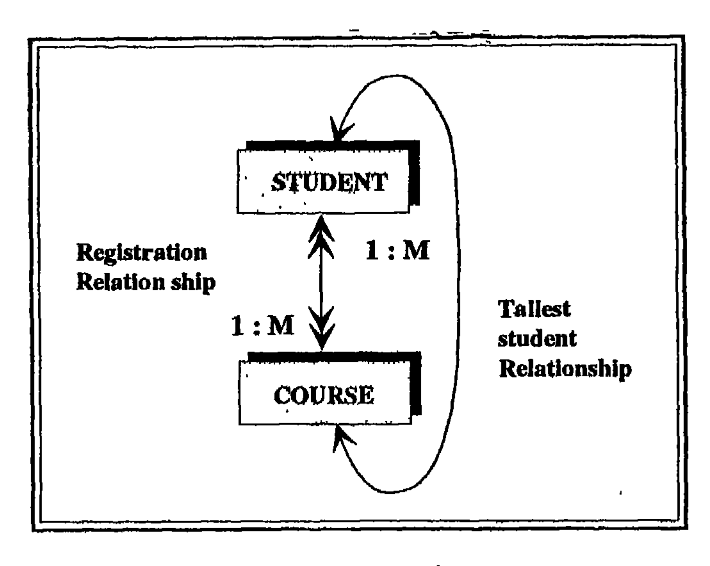
شكل (٣ - ١٨)

ويلاحظ أن علاقة الربط (1 : M) فى كلا الإتجاهين. وتسمى هذه الشبكة بالشبكة المعقدة (Complex Network). والشبكة البسيطة (Simple) هى التى لها العلاقة (1 : M) فى إتجاه واحد فقط كما فى شكل (٣ - ١٩)

وتسمح الشبكة بإمكانية المسار الدائرى (Cycle) كما فى شكل (٣ - ٢٠)

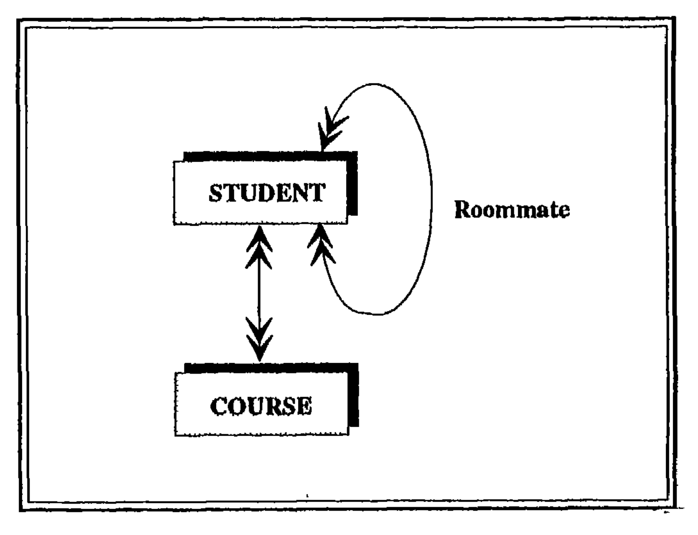


شكل (٣ - ١٩)



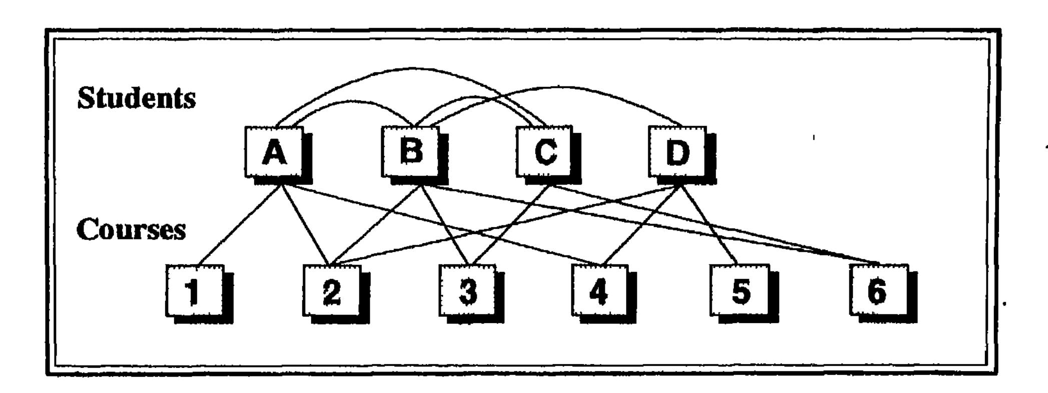
شكل (٣ - ٢٠)

وهنا نجد أن الطالب يمكن أن يسجل فى عدد من المواد. كما أن كل فصل به عدد معين من الطلبة بالإضافة إلى أن كل فصل يحوى أكثر الطلبة طولا وهكذا نجد أن هناك مسار دائرى من الطالب إلى الفصل إلى الطالب. أيضا تسمح الشبكة بالحلقات التكرارية (Loops). والحلقة التكرارية (Loop) هى علاقة بين نقطة الربط (Node) ونفسها كما فى شكل (٣ - ٢١)



شكل (٣ - ٢١)

فمثلا بعض الطلبة المسجلين لمواد معينة يمكن أن تجمعهم نفس الحلقة الدراسية (Roommate) بحيث تتكون حلقة تكرارية (Loop) حول نقطة الربط (Student). كما يتضح من الشكل (٣ - ٢٢).



شكل (٣ - ٢٢)

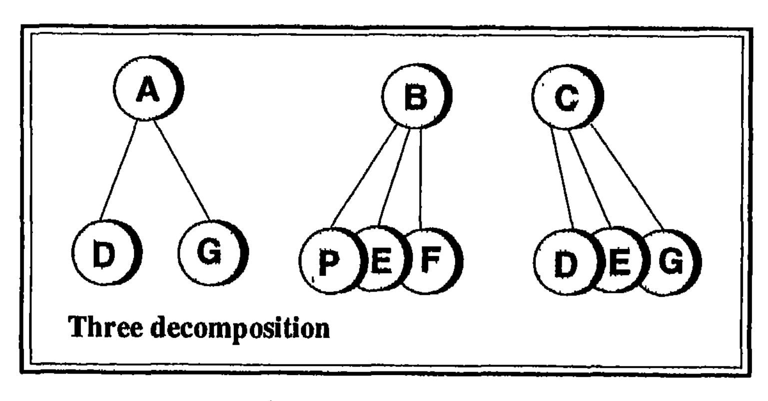
فمثلا الطالب (A) مسجل للمواد (4 , 2 , 1) ويقع فى نفس الحلقة الدراسية مع الطالب (B) المسجل للمواد (6 , 3 , 2).

٣ - ٣ - ٢ التمثيل الطبيعى (Physical Representation)

هناك طريقتان تستخدمان للتمثيل الطبيعى للشبكات مشابهة للمستخدمه مع الأشجار (Trees) ولكن أكثر تعقيدا. الطريقة الأولى باستخدام القائمة المتتابعة والثانية بإستخدام القائمة المرتبطة.

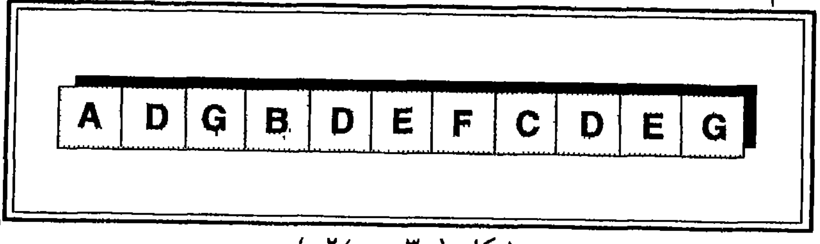
أ - القائمة المتتابعة (Sequential List)

وفى هذه الطريقة تستخدم قائمة متتابعة لنقاط الربط (Nodes) تبعا لنظام معين فمثلا الشبكة الموضحة فى شكل (٣ - ٢٢) يمكن تحليلها إلى تراكيب شجرية (Tree Structures) كما يتضح من الشكل (٣ - ٢٣).



شكل (٣ - ٢٣)

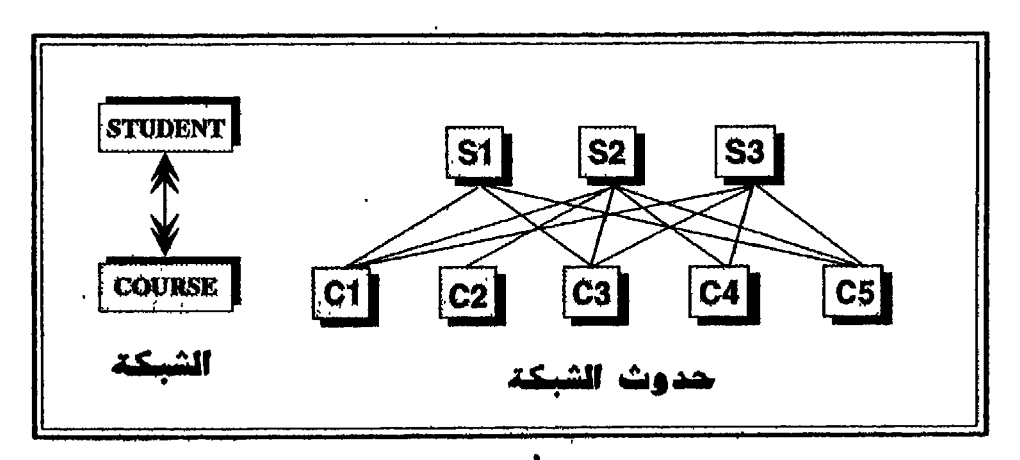
ويلاحظ من الشكل تكرار بعض نقاط الربط مثل النقطة (D) فهى تتكرر ثلاثة مرات وهذا التكرار صفة مقبولة من صفات عملية التحليل. ويلى عملية التحليل إستخدام القائمة المتتابعة لتمثيل التراكيب الشجرية والشكل (٣ - ٢٤) يوضح إستخدام القائمة المتتابعة لتمثيل الشبكة.



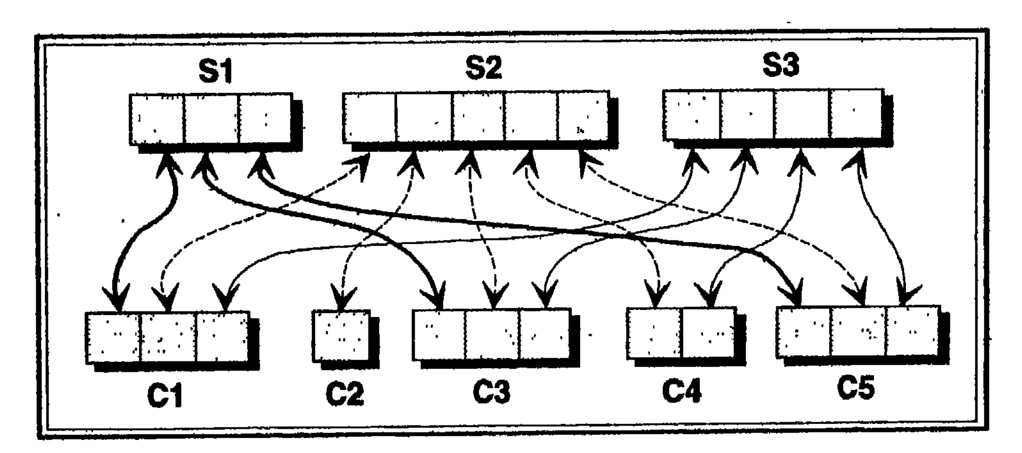
شكل (٣ - ٢٤)

ب - القائمة المرتبطة (Linked List)

تستخدم القوائم المرتبطة للتمثيل الفعلى للشبكات بطرق عديدة. تعتمد هذه الطرق على إستخدام مؤشرات (Pointers) متعددة عند كل نقطة ربط (Node) والتى لها علاقة ربط من واحد إلى كثيرين مع نقطة ربط أخرى. فمثلا علاقة الطالب بالمادة الموضحة فى شكل (٣ - ٢٥) يمكن تمثيلها كالموضح فى الشكل (٣ - ٢٦).



شكل (٣ - ٢٥)



شكل (٣ - ٢٦)

يلاحظ أن نقطة الربط (S1) مثلا لها ٣ مؤشرات والنقطة (S2) لها ٥ مؤشرات وهكذا. وكل نقطة ربط (Node) خاصة بالمادة سيكون لها عدد معين من

المؤشرات وهكذا يمكن تكوين قائمة مرتبطة كالموضحة فى شكل (٣ - ٢٧)

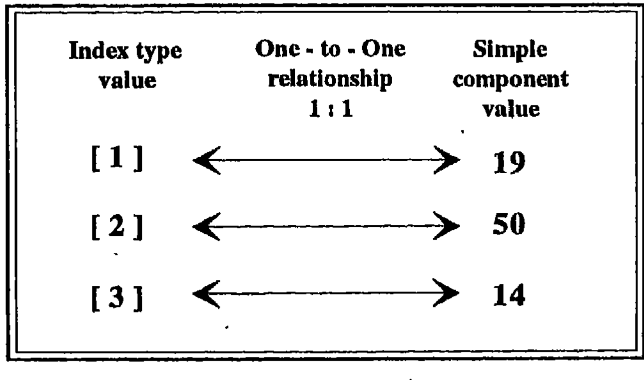
Record No.	Pointers			Overflow	Record
1	4	6	8	0	S1
2	4	5	6	9	S2
3	4	6	7	10	S3
4	1	2	3	0	C1
5	2	0	0	0	C2
6	1	2	0	0	C3
7	2	3	0	0	C4
8	2	2	0	0	C5

شكل (٣ - ٢٧)

وإذا كان لنقطة الربط (Node) أكثر من ثلاثة مؤشرات يوضع الزيادة فى الخانة الإحتياطية (Overflow Area).

٣ - ٤ المصفوفة (Array)

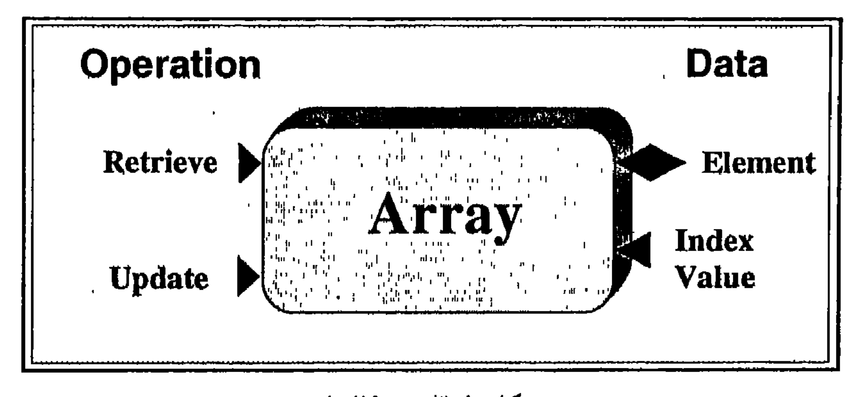
تعتبر المصفوفة من أهم وأقدم تراكيب البيانات وأكثرها إستعمالا وتتكون المصفوفة من مجموعة من العناصر الأولية (Atomic). (وفى بعض الأحوال الخاصة تكون العناصر مركبة (Structured)). وتجمع هذه العناصر صفة مشتركة وهى أنها بيانات أساسية (Fundamental Data) وترتبط منطقيا ببعضها بعلاقات خطية (Linear Relationships). ويمكن وصف أى مصفوفة بنوع عناصرها (Component Type) ونوع الفهرس (Index Type) فمثلا مصفوفة من النوع البسيط تحتوى على ثلاثة أرقام صحيحة (Integers) يقابل كل رقم منهم رقم فهرس معين كما يتضح من الشكل (٣ - ٢٨).



شكل (٣ - ٢٨)

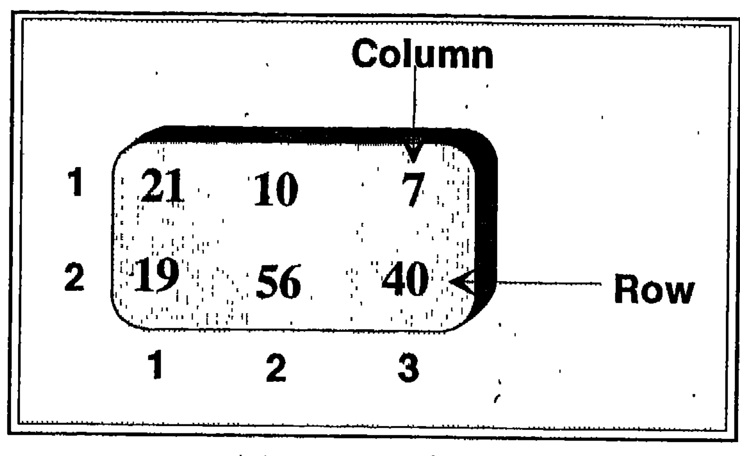
واضح أن العلاقة بين قيمة العنصر ورقم الفهرس علاقة واحد إلى واحد (1 : 1) ولايمكن أن تكون غير ذلك لان كل قيمة للفهرس يقابلها عنوان العنصر المقابل ولايمكن أن يشترك عنصران فى رقم فهرس واحد.

ومن العمليات التى يمكن إجراؤها على المصفوفة عملية التحديث (Update) والإستعادة (Retrieve) والشكل (٣ - ٢٩) يوضح المصفوفة فيما يسمى بشكل الكبسولة (Capsule - Like View).



شكل (٣ - ٢٩)

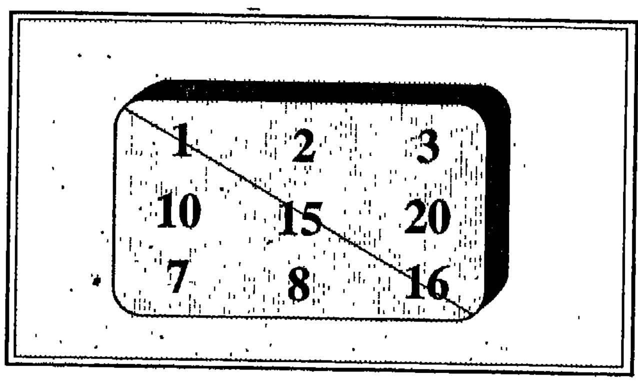
ويمكن أن يكون أحد عناصر المصفوفة مصفوفة بذاته وهو ما يجعل المصفوفة متعددة البعد (Multidimensional). والشكل (٣ - ٣٠) يوضح مصفوفة ثنائية البعد (Two - dimensional) وهى تتكون من صفين وثلاثة أعمدة.



شكل (٣ - ٣٠)

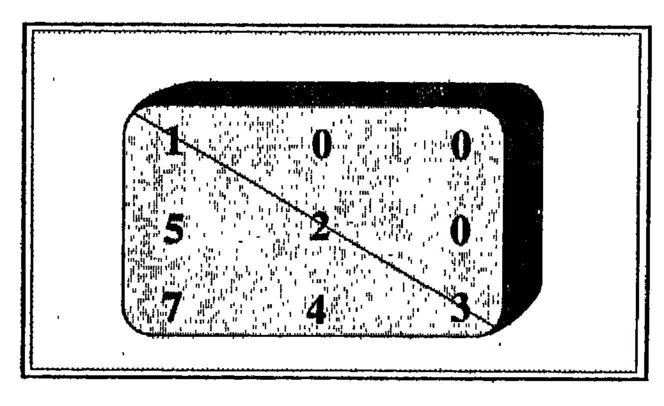
ويلاحظ أن هناك رقمين للفهرسة (1..2) و (1..3) بحيث يسهل تحديد عنوان أى عنصر فمثلا العنصر (56) عنوانه (2 , 2) أى الصف الثانى - العمود الثانى وهكذا أيضا يلاحظ العلاقة بين رقم الفهرس وقيمة العنصر من واحد إلى واحد.

وفى حالة تساوى عدد الصفوف والأعمدة تسمى المصفوفة مربعة (Square) شكل (٣ - ٣١)



شكل (٣ - ٣١)

وللمصفوفة المربعة وتر (Diagonal) يقسم المصفوفة إلى مثلثين متساويين مثلث علوى (Upper) وآخر سفلى (Lower) وفى حالة ما إذا كانت عناصر المثلث العلوى تساوى أصفار تسمى المصفوفة بالمصفوفة ذات المثلث السفلى (Lower Triangle) كما فى شكل (٣ - ٣٢).

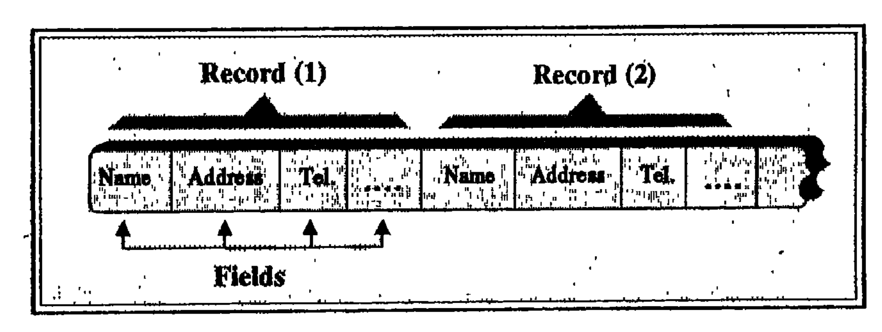


شكل (٣ - ٣٢)

وعكس ذلك تسمى المصفوفة ذات المثلث العلوى (Upper - Traingle).

٣ - ٥ السجـــلات (Records)

من أهم ما يميز السجلات هو قدرتها على أن تضم وحدات بيانات مختلفة الأنواع فى كيان واحد. وهذه الميزة تسمح برؤية مجموعة من البيانات عن شئ معين ككيان منطقى واحد (Single Logical Entity) فمثلا السجل الخاص بشخص معين يمكن أن يحتوى على حقول مثل الإسم والعنوان ورقم التليفون والنوع و ... إلخ أنظر شكل (٣ - ٣٣). ويمكن إعتبار هذه البيانات وحدة واحدة (Single Element) وبالتالى يمكن تخزينها أو إستعادتها أو نقلها وأيضا يمكن إستخدامها منفصلة. ويتكون السجل من مجموعة من الحقول (Fields) وكل حقل يحتوى على وحدات بيانات أولية (Atomic) أو مركبة (Structured).



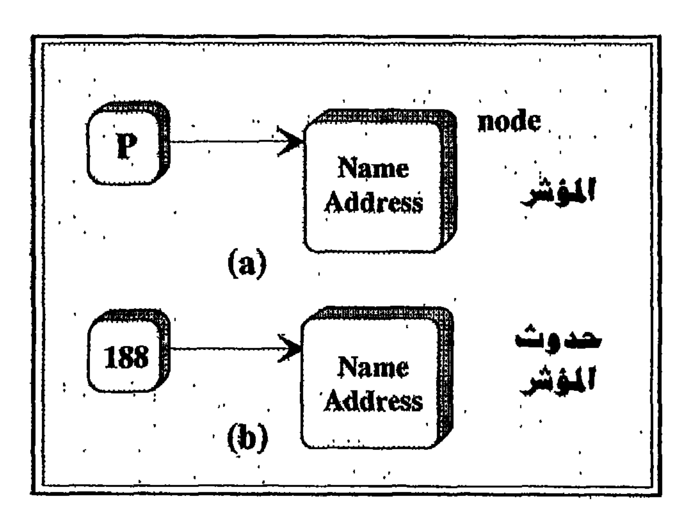
شكل (٣ - ٣٣)

ويعرف كل حقل برقم الحقل (Field No.) وإسم الحقل (Identifier) ونوع محتوياته (Data Types) وطول الحقل (Field Length) كما سبق شرحه.

٣ - ٦ المؤشـــرات (Pointers)

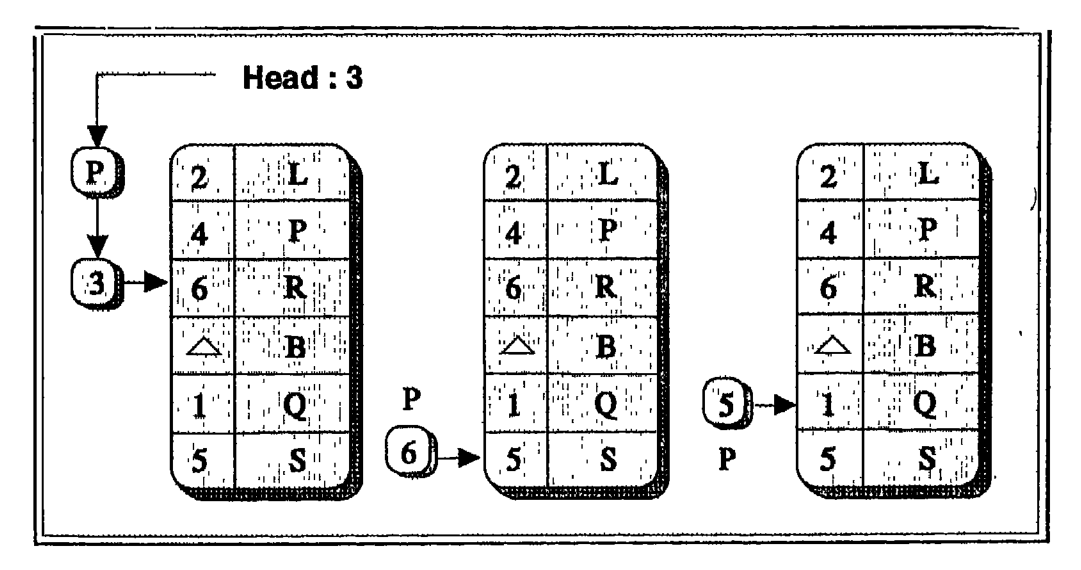
المؤشر (Pointer) هو نوع بيانات (Data Type) قيمته هى عناوين أنواع أخرى من البيانات. على سبيل المثال إذا كانت البيانات المراد البحث عنها تقع فى الصف الخامس فى قائمة معينة فإن قيمة المؤشر تكون خمسة. ويأخذ المؤشر قيما تتراوح من صفر (0) إلى (M - 1) حيث (M) هو عدد البايت (Bytes) فى الذاكرة.

والشكل (٣ - ٣٤) يوضح نقطة ربط معينة (Node) والمؤشر ، وهنا نجد أن قيمة المؤشر هى عنوان الذاكرة (Memory Address) لأول بايت فى البيانات الموجودة فى هذه النقطة وهو (188).



شكل (٣ - ٣٤)

مثال آخر سبق إيضاحه فى شكل (٣ - ٤) موضح فى شكل (٣ - ٣٥) باستخدام أسلوب المؤشرات (Pointers).



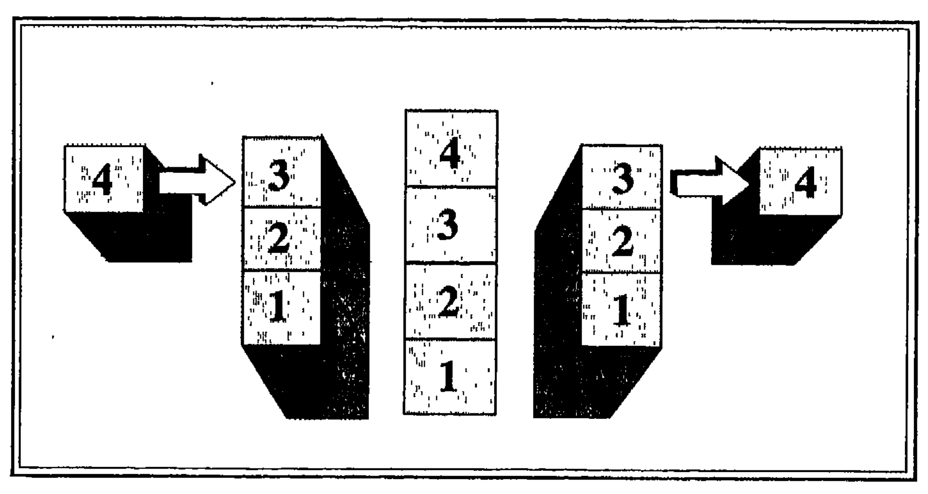
شكل (٣ - ٣٥)

ويلاحظ فى القائمة (١) أن المؤشر يأخذ قيمة رأس القائمة (Head) ويقف عند السطر الثالث أى السطر (R) ثم يأخذ القيمة (6) وينتقل إلى السطر السادس ويقرأ العنصر (S) ثم يأخذ القيمة (1) وهكذا حتى يأخذ القيمة (Δ) أى يقف عند نهاية القائمة وبهذا نحصل على الترتيب الطبيعى (R - S - Q - L - P - B).

٣ - ٧ الرصـــة (Stack)

تعرف الرصة على أنها قائمة البيانات (Data List) يتم تحديد صفاتها الرئيسية عن طريق القواعد التى تحكم عمليات الحشر (Insertion) أو الحذف (Deletion) لعناصرها. والعنصر الذى يمكن حذفه أو تحريكه (Removed) هو أحدث عنصر تم حشره فيما يعرف بقاعدة الداخل أخيرا يخرج أولا(Last in / First out) أو باختصار (LIFO) شكل (٣ - ٣٦) وتسمى عملية إضافة عنصر جديد للرصة بالدفع (Push). وعملية حذف أحدث عنصر أو تحريكه بالخروج (Pop).

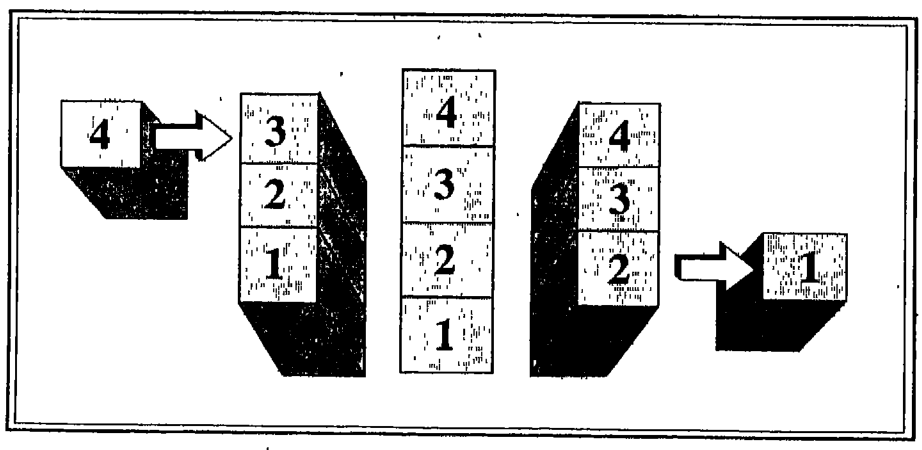
فمثلا لإيجاد قيمة العميلة الحسابية ((7 + 5) x 10) باستخدام الرصة أولا ندفع (Push) (5) ثم (7) ثم نجمع ثم ندفع (10) ثم نضرب وتقوم عمليات الجمع والضرب أولا بإخراج (Pop) العناصر المطلوبة من الرصة وأخيرا يتم دفع النتيجة داخل الرصة.



شكل (٣ - ٣٦)

٣ - ٨ الطابـــور (Queue)

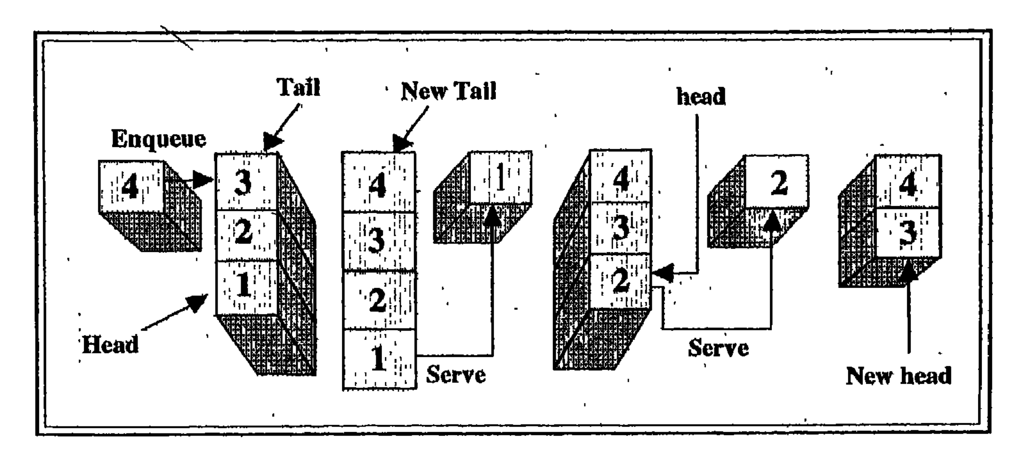
يعتبر الطابور من أعلى مستويات تمثيل البيانات المختصر (Abstract Data) وله أهمية قصوى فى العمليات الحسابية. والطابور من الأشياء اليومية المألوفة فى حياتنا مثل الطابور فى البنك أمام شباك الصرف أو فى السينما أمام شباك التذاكر. ويحكم الطابور الخاص بالبيانات قاعدة الداخل أولا يخرج أولا (First in / First out) أو بإختصار (FIFO) كما هو موضح فى شكل (٣ - ٣٧).



شكل (٣ - ٣٧)

وتسمى عملية إضافة عنصر جديد للطابور دخول الطابور (Enqueue) وعملية الحذف

أو التحريك بالخدمة (Serve). ويمكن أن يكون الطابور جزءا من مصفوفة كما يتضح من الشكل (٣ - ٣٨).



شكل (٣ - ٣٨)

الفصل الرابع

الفرز والبحث

(Sorting And Searching)



كثيرا ما نحتاج إلى ترتيب السجلات بترتيب معين مخالف للترتيب الذى أدخلت به إلى الحاسب أول مرة. فمثلا فى قاعدة بيانات المكتبة نحتاج إلى ترتيب السجلات هجائيا حسب إسم المؤلف حتى يسهل البحث عن مؤلف معين. ومن الطرق شائعة الإستعمال فى ترتيب السجلات طريقة الفرز (Sorting).

٤ - ١ الفــــرز (Sorting)

إن لعملية ترتيب البيانات أكبر الأثر فى عملية البحث عنها. فالبيانات غير المرتبة يتم البحث خلالها بطريقة تتابعية (Sequential) من البداية للنهاية فى حين يكون أسلوب البحث بسيطا فى حالة البيانات المرتبة وتهتم عملية الفرز (Sorting) بترتيب البيانات طبقا لنظام معين (مثلا ترتيب أبجدى تصاعدى أو تنازلى). وتنقسم عملية الفرز إلى فرز خارجى (External Sorting) وفرز داخلى (Internal Sorting). الفرز الداخلى يكون فى حالة ما إذا كانت كمية البيانات المطلوب فرزها صغيرة بحيث تتم عملية الفرز فى ذاكرة الوصول العشوائى للكمبيوتر (Random Access Memory). أما إذا كانت كمية البيانات المطلوب فرزها كبيرة لزم لذلك إستخدام طريقة الفرز الخارجى بعد تخزين البيانات على وسيط تخزين ثانوى. وفى هذا الجزء سنهتم بالفرز الداخلى. وتحدد طريقة الفرز الداخلى حجم العمليات المطلوب لعملية الفرز والمقصود بها عملية مقارنة وحدتين للبيانات وعملية تحريك وحدة بيانات من مكان إلى مكان آخر.

ومن طرق الفرز الداخلى المعروفة الفرز البسيط (Simple Sort) والفرز المتقدم (Advanced Sort) وفرز الجذر (Root Sort). على سبيل المثال فإن ملف الموظفين المبين فى شكل (٤ - ١) يحتوى على مجموعة من السجلات وكل سجل به أربعة حقول (رقم الموظف - إسم الموظف - القسم - المرتب) بالإضافة إلى رقم كل سجل فى القائمة والذى لا يمكن فرزه كجزء من السجل.

وتتم علمية الفرز عن طريق أحد الحقول ويسمى حقل الفرز (Sort Field) ففى حالة فرز هذا الملف باعتبار أن حقل المرتب (Salary) هو حقل الفرز نحصل على القائمة الموضحة فى شكل (٤ - ٢)

Element position	Employee No.	Name	Department	Salary
1	005	Aly	Hardware	72
2	010	Ahmed	Language	40
3	036	Omar	Programming	74
4	049	Baker	Hardware	69

شكل (٤ - ١)

Element position	Employee No.	Name	Department	Salary
1	010	Ahmed	Language	40
2	049	Baker	Hardware	69
3	005	Aly	Hardware	72
4	036	Omar	Programming	74

شكل (٤ - ٢)

ويتضح من الشكل أن عملية الفرز تمت بترتيب تصاعدي كذلك يلاحظ تغيير أماكن السجلات المختلفة ورقم كل عنصر (Element Position). ويمكن أن تتم عملية الفرز على حقل المرتب فقط لتجنب تحريك كل وحدات البيانات فى السجلات المختلفة كما يبين شكل (٤ - ٣)

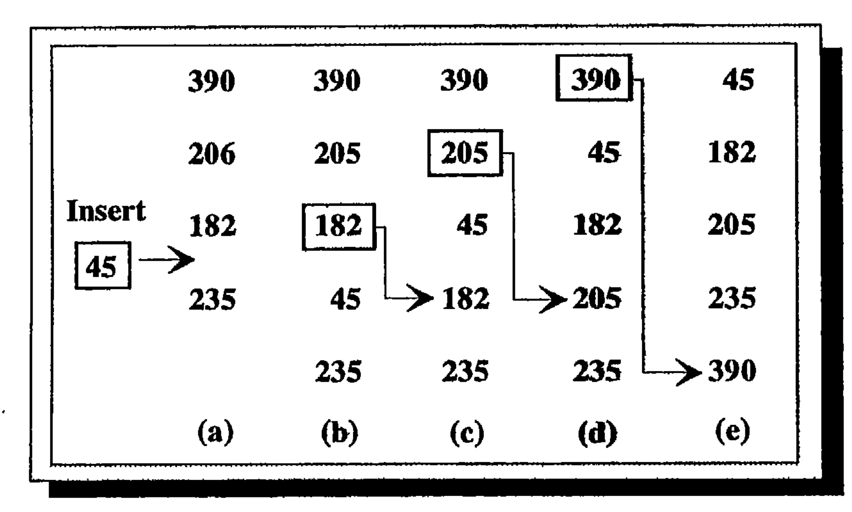
Salary	Element position
40	2
69	4
72	1
74	3

شكل (٤ - ٣)

ويلاحظ أن مكان العنصر ثابت ولم يتغير ويستعمل كمؤشر (Pointer) للدلالة على مكان التخزين لكل سجل ويسمى الشكل (٤ - ٣) بمصفوفة الفهرس(Index Array).

٤ - ٢ فرز الإضافة (Insertion Sort)

يعنى فرز الاضافة إضافة وحدة بيانات إلى مجموعة من الوحدات التى تم فرزها بحيث يكون التتابع الناتج مفروزا أيضا. ويوضح الشكل (٤ - ٤) طريقة فرز أربعة أعداد صحيحة.

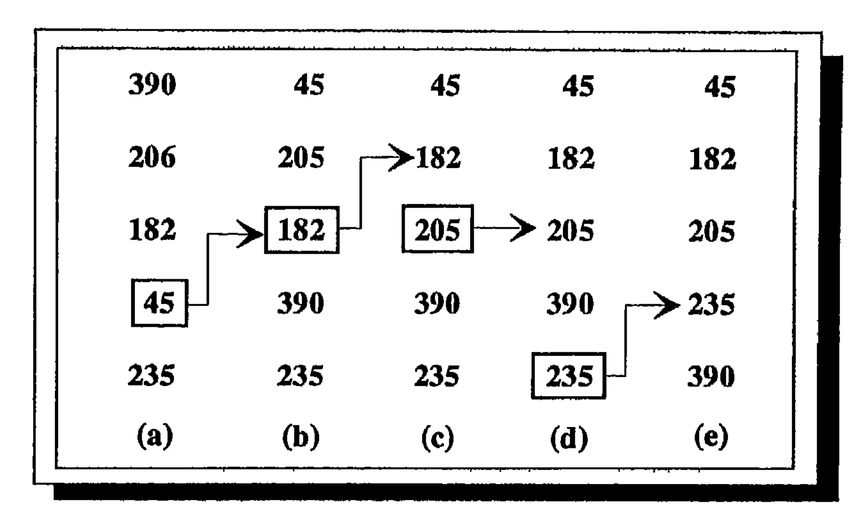


شكل (٤ - ٤)

ويوضح العمود الأول (a) مصفوفة البيانات التى تم إدخالها. ثم يتم حشر العدد (45) فى المكان الموضح. ويلى ذلك مجموعة من الانتقالات للأعداد بهدف استعادة الترتيب التصاعدى للمصفوفة والذى يتضح من العمود (e).

٤ - ٣ فرز الإختيار (Selection Sort)

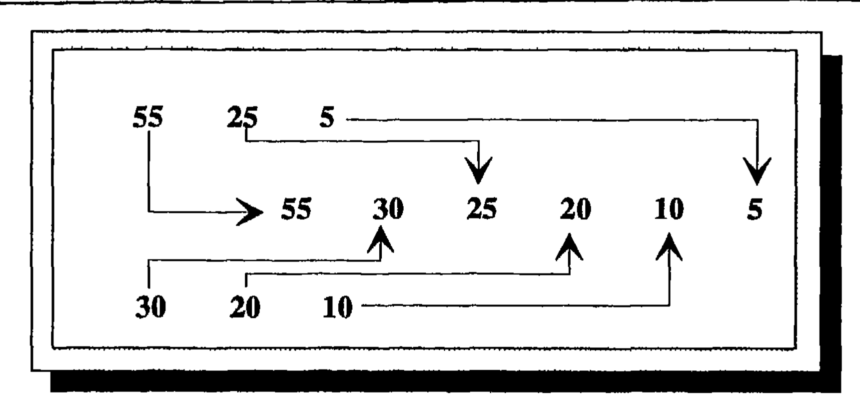
فرز الإختيار هو إختيار أصغر (أو أكبر) وحدة بيانات من بين مجموعة من وحدات البيانات. والشكل (٤ - ٥) يوضح عملية فرز خمسة أرقام صحيحة لتحديد أصغرها. ويلاحظ فى العمود (a) تحديد العدد (45) كأصغر قيمة وبالتالى يتم نقله الى أول العمود. وفى العمود (b) يتم تحديد العدد (182) كأصغر قيمة وبالتالى يتم نقله الى أول العمود وهكذا حتى نصل الى الترتيب النهائى فى العمود (e).



شكل (٤ - ٥)

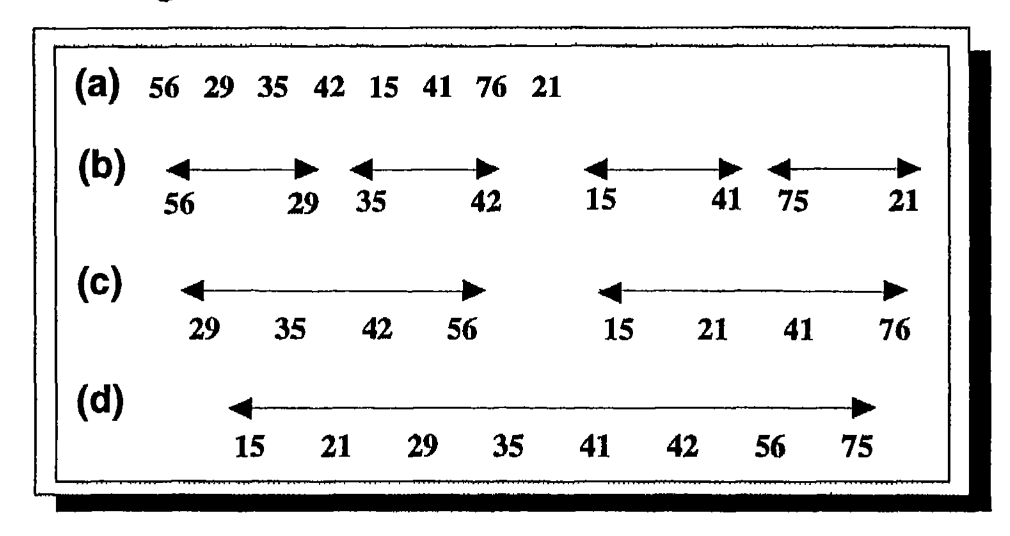
٤ - ٤ فــرز الدمج (Merge Sort)

فى هذا النوع من الفرز يتم دمج قائمتين فرعيتين (SubLists) كلا منهما تم فرزه من قبل فى قائمة واحدة مفروزة. وتتم عملية الدمج بمقارنة كل وحدة بيانات من القائمة الأولى بوحدة بيانات من القائمة الثانية ويتم ترتيب الوحدتين (ترتيب تنازلى مثلا) بحيث تحل إحداهما مكان وحدة البيانات الثانية فى القائمة وهكذا حتى تتم عملية الفرز. والشكل (٤ - ٦) يوضح الفرز بالدمج.



شكل (٤ - ٦)

ونرى هنا سهولة عملية الفرز لأن القوائم الفرعية تم فرزها من قبل أما فى حالة ما إذا كانت القوائم الفرعية غير مرتبة فإننا نعامل كل وحدة بيانات على أنها قائمة فرعية مستقلة يتم فرزها ودمجها فى قائمة مكونة من وحدتين ثم تكرر العملية لتكوين قائمة من أربع وحدات وهكذا حتى تنتهى عملية الفرز. والشكل (٤ - ٧) يوضح هذه الطريقة.



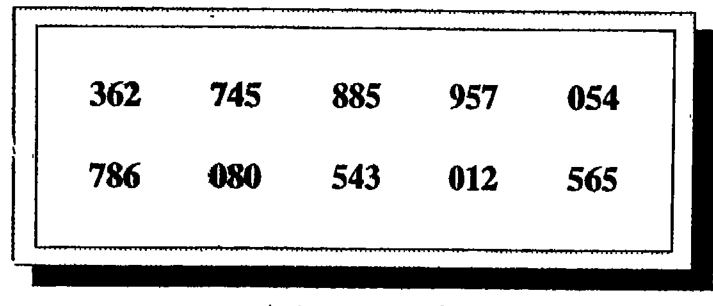
شكل (٤ - ٧)

٤ - ٥ فــرز الجـــذر (Radix Sort)

من الصفات المميزة لطرق الفرز السابقة أنها جميعا تعتمد على المقارنة (Comparing) والتحريك (Moving) لوحدات البيانات المراد فرزها. أما طريقة فرز

الجذر (Radix Sort) فلا يتم فيها أى مقارنة ولا تحريك ولكنها تعتمد على ترتيب وحدات البيانات عدة مرات تتوقف على عدد خانات هذه الأعداد.

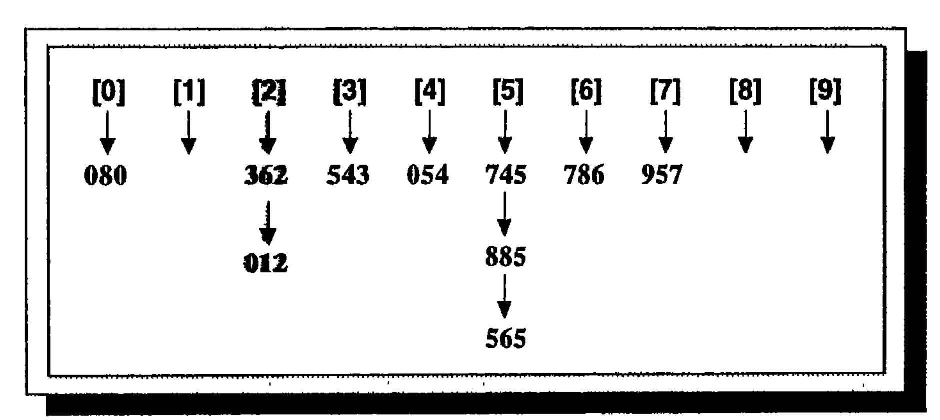
فمثلا الشكل (٤ - ٨) يوضح عملية فرز مجموعة من العناصر بطريقة الجذر.



362	745	885	957	054
786	080	543	012	565

شكل (٤ - ٨)

ويلاحظ أن كل عنصر يحتوى على ثلاثة خانات رقمية (Digits) ولهذا نضع (R = 3). والخطوة الأولى هى ترتيب الأعداد بناء على رقم الآحاد كما يتضح من الشكل (٤ - ٩).



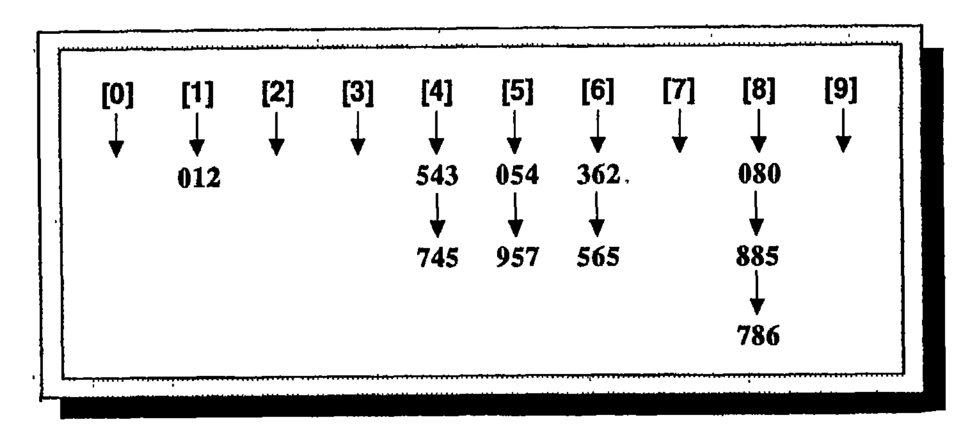
شكل (٤ - ٩)

ثم توضع القوائم الموجودة فى شكل (٤ - ٩) فى مجموعة واحدة لتكون القائمة الموجودة فى شكل (٤ - ١٠).

080	362	012	543	054
745	885	565	786	957

شكل (٤ - ١٠)

ويعاد ترتيب العناصر حسب رقم العشرات كما يتضح من الشكل (٤ - ١١).



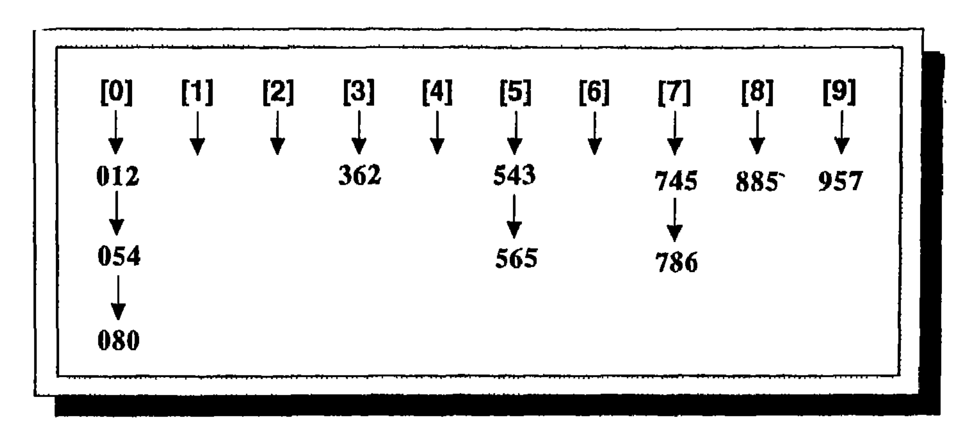
شكل (٤ - ١١)

ثم يتم تكوين القائمة الموجودة فى شكل (٤ - ١٢)

012	543	745	054	957
362	565	080	885	786

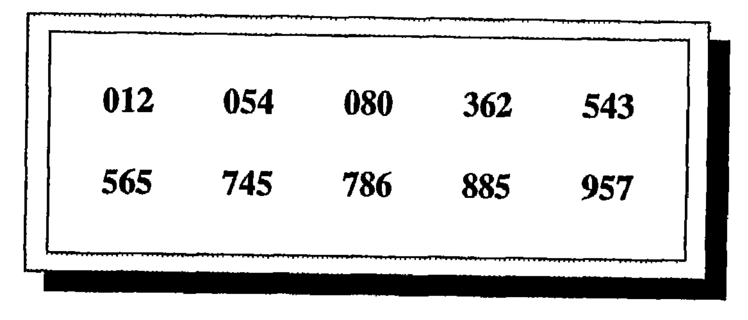
شكل (٤ - ١٢)

وأخيرا يعاد ربط كل عنصر على حسب رقم المئات كما يتضح من الشكل (٤ - ١٣)



شكل (٤ - ١٣)

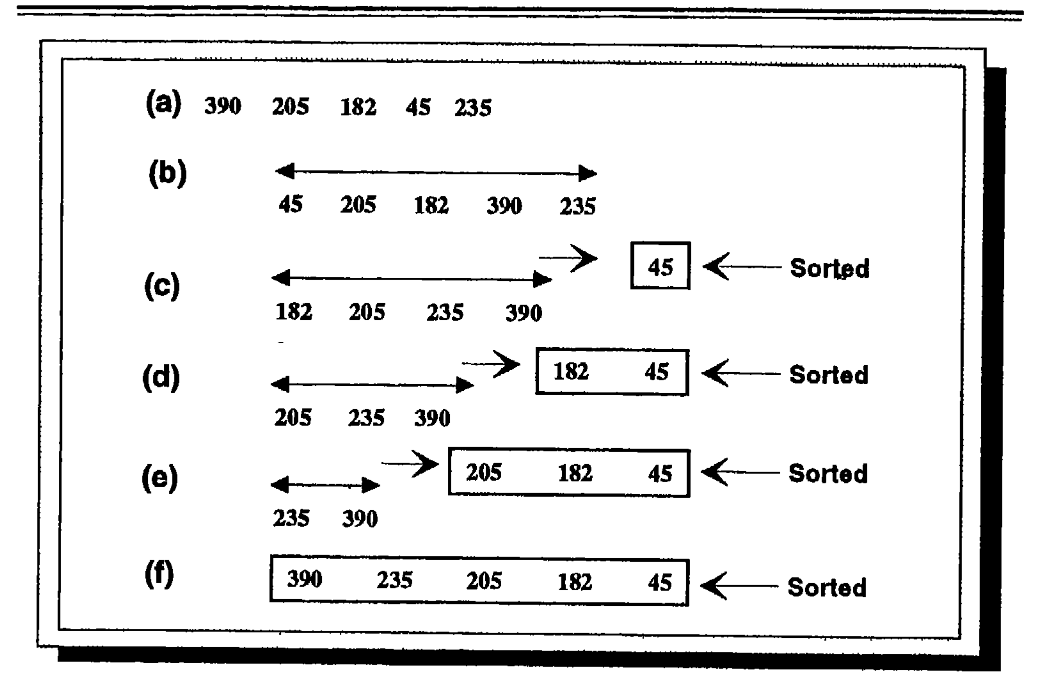
لنحصل فى النهاية على قائمة مرتبة كما فى شكل (٤ - ١٤)



شكل (٤ - ١٤)

٤ - ٦ فـرز الحـزم (Heap Sort)

يتم فرز الحزم على مرحلتين ، المرحلة الأولى وفيها توضح المعلومات فى حزمة ثم تليها المرحلة الثانية وهى إستخراج البيانات من الحزمة فى صورة مرتبة والمثال التالى يوضح هذه المراحل. أنظر شكل (٤ - ١٥)



شكل (٤ - ١٥)

ويلاحظ أنه أثناء كل مرحلة من مراحل الفرز أن أصغر العناصر يتم تحريكه من أعلى الحزمة ووضعه فى القائمة التى تم فرزها (Sorted) وبقية العناصر يعاد بناؤها فى حزمة جديدة وهكذا حتى تنتهى عملية الفرز.



الفصل الخامس

تصميم قاعدة البيانات



تصميم قاعدة البيانات شأنه شأن أى عملية تصميم أخرى يمر بمراحل عديدة تنتهى بمرحلة التصميم الفعلى (Physical Design). وهذه المرحلة يتم فيها تحديد شكل السجل المخزن (Record Format) وطرق التعامل مع السجلات (Access Method) وطرق تأمين قاعدة البيانات (Security) بالإضافة إلى عمل النسخ الإحتياطية (Backups).

ومن أهم مراحل التصميم المنطقى للنظام (Logical Design) مرحلة تطبيع البيانات (Normalization) وخاصة بالنسبة لقواعد البيانات العلاقية(Relational Databases). وفى هذا الفصل سوف نتناول بالدراسة كل ما يتعلق بمرحلة تطبيع البيانات.

٥ - ١ تطبيع البيانات (Normalization)

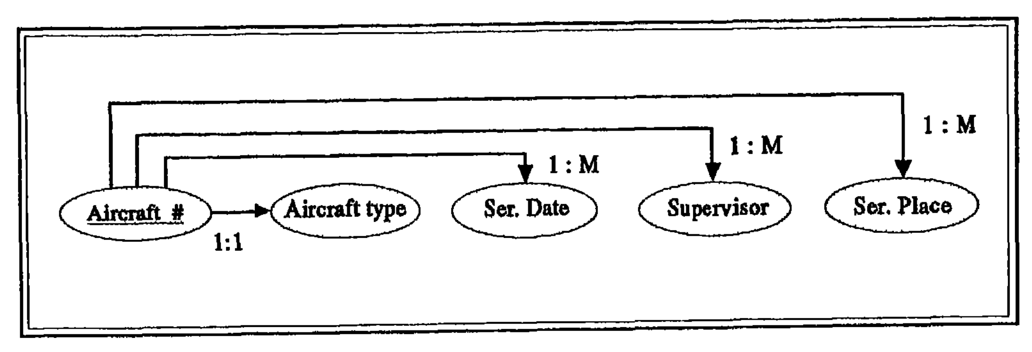
كما سبق الإيضاح فإن كل سجل يحتوى على عدد من الحقول التى تختص بمجموعة من الكيانات (Entities). فمثلا فى قاعدة بيانات صيانة الطائرات (Maintenance) لإحدى شركات الطيران تحتوى السجلات على حقول رقم الطائرة (Aircraft #) ونوعها وتاريخ الصيانة (Date Serviced) ونوع الصيانة (Type of Service) ومشرف الصيانة (Supervisor). ويلاحظ أن السجل يهتم بثلاثة كيانات وهم الطائرة (Aircraft) والصيانة (Service) والمشرف (Supervisor). فعند تمثيل النموذج المنطقى لهذا السجل يصبح كالموضح فى شكل (٥ - ١).

Aircraft #	Aircraft Type	Service Date	Service Type	Super-visor	Service place
475310	747	11 / 89 3 / 90 12 / 91	Engine overhaul Seat repair Navigation system	Aly Ahamed Lutfy	F 103 H 04 H 10
304168	DC10	2 / 91 10/ 91	Glass replacement Engine Overhaul	Khaled Aly	H 07 H 03

شكل (٥ - ١)

ويلاحظ من الشكل أن هناك عدة قيم لنفس الكيان فى بعض الأعمدة حيث أن

الطائرة يمكن أن تتم لها أكثر من عملية صيانة لذلك تعتبر بيانات الصيانة (Service) أحد المجموعات المتكررة (Repeating Groups). وهذه المجموعات المتكررة تؤثر على تكامل قاعدة البيانات (Integrity) وكفاءة عمليات البحث. فالبحث كما سبق الإيضاح يتم بواسطة حقل المفتاح (Key Field) والذى يجب أن يكون منفردا (Unique) أى يمثل سجلا محددا وبيانات محددة لكل حقل فى السجل. ولكن البيانات بالصورة السابق ذكرها فى شكل (٥ - ١) لاتحتوى على حقل مفتاح منفرد. فمثلا بالإعتماد على رقم الطائرة (Aircraft #) كحقل مفتاح وباستخدام أسلوب الربط بين الحقول الموضح فى الجزء (٢ - ٥) فإن العلاقة بين رقم الطائرة وباقى الحقول تكون كالموضحة فى شكل (٥ - ٢).



شكل (٥ - ٢)

ويلاحظ من الشكل أن العلاقة بين رقم الطائرة ونوع الطائرة هى علاقة واحد إلى واحد (1 : 1) بينما العلاقة بين رقم الطائرة وباقى الحقول هى علاقة واحد إلى كثيرين (1 : M) وهذا يعنى أن حقل رقم الطائرة لايمكن استخدامه كحقل مفتاح منفرد (Unique) للبحث عن سجل معين. كما يلاحظ أن هناك بيانات متكررة فى أكثر من موضع فمثلا البيانات الخاصة بصيانة المحرك (Engine Overhaul) موجودة فى أكثر من سجل فعندما يراد تغيير أحد البيانات المتكررة فإن ذلك يتطلب البحث عن كل السجلات التى تحتوى على هذا البيان المتكرر وإذا لم يتم تغيير هذا البيان فى أحد السجلات فإن ذلك سوف يؤثر على تكامل قاعدة البيانات (Database Integrity).

ويطلق على هذا النموذج " النموذج الغير مطبع " (Unnormalized) وهناك عدة مراحل لتطبيع هذا النموذج حتى يصل إلى الصورة التى يستطيع الحاسب التعامل معها والأجزاء الآتية توضح مراحل تطبيع هذا النموذج.

٥ - ١ - ١ نموذج التطبيع الأول (First Normal Form)

هذا النموذج يتخلص من البيانات المتكررة بحيث يكون كل عمود محتويا على قيمة واحدة لكل سجل وذلك بنقل المجموعات المتكررة فى علاقات (Relations) جديدة.

ويستخدم نموذج التطبيع الأول لتطبيع النموذج المنطقى للسجل الموضح فى شكل (٥ - ١) كالآتى :

١ - يتم فصل علاقة الطائرة (Aircraft) والتى تحتوى على الحقول غير المتكررة مثل رقم الطائرة (# Aircraft) ونوعها (Type) ويستخدم حقل رقم الطائرة كحقل مفتاح لهذه العلاقة كما فى شكل (٥ - ٣)

Aircaft #	Aircraft type
475310	747
304168	DC 10
18476	737
.....	....

شكل (٥ - ٣)

٢ - يتم فصل علاقة الطائرة والخدمة (Aircraft - Service) والتى تحتوى على مجموعة الحقول المتكررة مثل (Service Date) ، (Service Type) ، (Supervisor) ويستخدم حقل مفتاح مركب (Composite Key) لهذه المجموعة يتكون من الجمع بين حقل رقم الطائرة (# Aircraft) وتاريخ الصيانة (Service Date). أنظر شكل (٥ - ٤).

ويلاحظ أن رقم ونوع الطائرة ضروريان لتحديد مشرف الصيانة (Supervisor). ومن عيوب نموذج التطبيع الأول (First Normal Form) ظهور عدة مشكلات يمكن تلخيصها فى الآتى :

Aircraft #	Service Date	Service Type	Supervisor	Service place
475310	11 / 89	Engine overhoul	Aly	F 03
475310	3 / 90	Seat repair	Ahamed	H 07
475310	12 / 91	Navigation system	Lutfy	H 10
475310	2 / 91	Glass replacement	Khaled	H 05
475310	10 / 91	Engine Overhoul	Aly	H 03

شكل (٥ - ٤)

أ - مشكلة الإضافة (Insertion)

نفرض أنه يراد إضافة تاريخ جديد سوف تتم فيه عملية صيانة فإن هذه الإضافة لايمكن أن تتم إلا بعد تحديد رقم الطائرة وذلك لأن حقل رقم الطائرة (Aircraft #) يعتبر جزءاً من الحقل المركب (Composite Key) المستخدم كمفتاح للعلاقة.

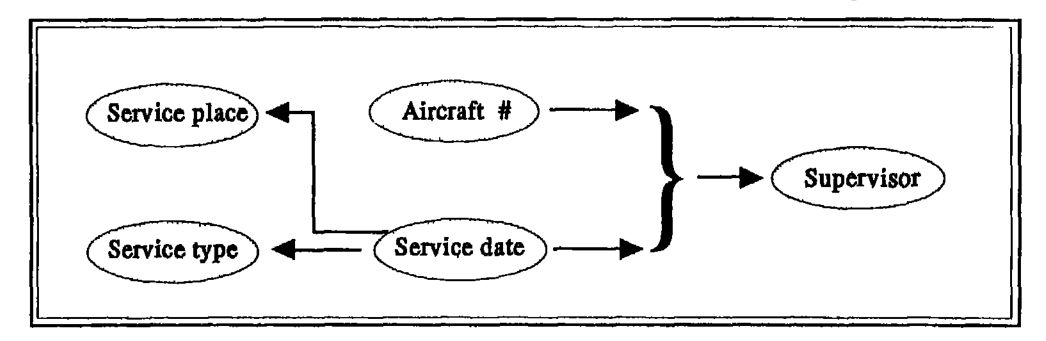
ب - مشكلة المسح (Deletion)

عند حذف بيانات إحدى الطائرات والتى أصبحت خارج الخدمة يتم حذف بيانات الصيانة ومشرف الصيانة وبالتالى تختفى هذه البيانات من قاعدة البيانات ولايمكن الرجوع إليها.

جـ - مشكلة التحديث (Update)

وتظهر هذه المشكلة عند تعديل أحد البيانات المتكررة فى أكثر من سجل. فى هذه الحالة نحتاج إلى البحث عن كل تكرار للبيان واستبداله.

وسبب هذه المشكلات الثلاثة أن هناك بعض الحقول التى ترتبط بحقل تاريخ الصيانة (Service Date) فقط وليس الحقل المركب (Aircarft - Service Date) كما يتضح من الشكل (٥ - ٥)



شكل (٥ - ٥)

وتسمى الحقول التى تعتمد على جزء من الحقل المركب ، مثل حقل (Service Type) بالحقول المعتمدة جزئيا على حقل المفتاح (Partially Dependent) وهى التى يجب التخلص منها للوصول إلى النوع الجديد من التطبيع وهو نموذج التطبيع الثانى (Second Normal Form).

٥ - ١ - ٢ نموذج التطبيع الثانى (Second Normal Form)

يستخدم هذه النموذج ، كما سبق الإيضاح ، فى التخلص من الحقول المعتمدة جزئيا (Partially Dependent) على حقل المفتاح وذلك بتقسيمها إلى علاقتين إحداهما تحتوى على الحقول التى تعتمد إعتمادا كليا على حقل المفتاح والأخرى تحتوى على الحقول التى تعتمد إعتمادا جزئيا عليه ، أنظر شكل (٥ - ٦).

ويلاحظ من الشكل أن العلاقة (Aircraft - Service) قد تم تقسيمها إلى علاقتين كالآتى :

أ - علاقة المشرف (Supervisor) التى تحتوى على الحقل المركب رقم الطائرة - تاريخ الصيانة (Aircraft # - Service Date) وهو يعتمد إعتمادا كليا على الحقل المركب.

Aircraft #	Date Service	Type of Service	Supervisor	Place

Supervisor

Aircraft #	Service Date	Supervisor
475310	11 / 89	Aly
475310	3 / 90	Ahamed
475310	12 / 91	Lutfy
304168	2 / 91	Khaled
304168	10 / 91	Aly

Date - Type

Service Date	Service Type
11 / 89	Engine overhoul
3 / 90	Seat repair
12 / 91	Navigation system
2 / 91	Glass replacement
10 / 91	Engine Overhoul

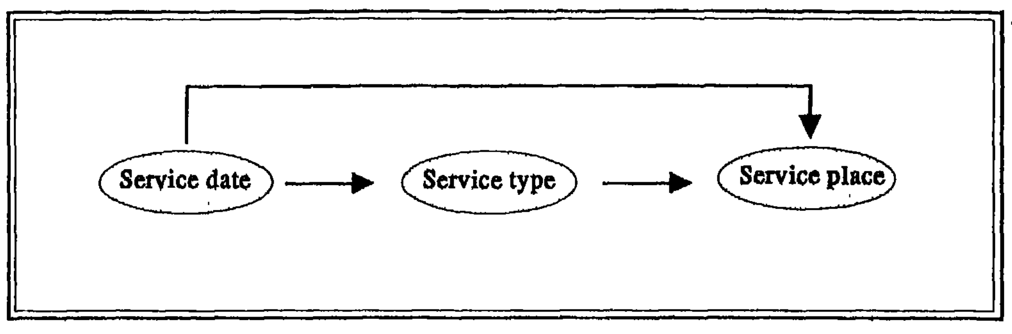
شكل (٥ - ٦)

ب - علاقة (Date - Type) التى تعتمد على تاريخ الصيانة (Service Date) كحقل مفتاح (Key Field) بالإضافة إلى باقى الحقول التى تعتمد على حقل تاريخ الصيانة.

وباستخدام نموذج التطبيع الثانى فإن المشاكل السابق شرحها فى نموذج التطبيع الأول قد أمكن التغلب عليها. حيث أصبح كل تاريخ صيانة معرف مرة واحدة فى العلاقة (Date - Type) وبالتالى أصبح أى تحديث أو تعديل محصورا فى سجل واحد كما أن فصل بيانات الطائرة (Aircraft) عن بيانات تاريخ الصيانة جعل من السهل إضافة أو حذف أى تاريخ صيانة دون الحاجة إلى بيانات الطائرة. ورغم ذلك فإن العلاقة (Date - Type) تحتاج إلى مزيد من التطبيع وذلك لان هناك علاقة متعدية (Transitive) بين نوع الصيانة ومكان الصيانة (Service Place) والشكل (٥ - ٧) يوضح العلاقات بين حقول هذه العلاقة.

ويلاحظ أن كل حقل يعتمد على حقل المفتاح (Service Date) بينما هناك حقل (Service Place) يعتمد على حقل (Service Type) بالإضافة إلى حقل المفتاح وذلك لأن كل نوع صيانة له مكان محدد وهذا يؤدى إلى تكرار نوع الصيانة فى أكثر

من سجل وهنا تظهر نفس المشاكل التى سبق شرحها والخاصة بعملية مسح أو إضافة أو تحديث البيانات. لذلك يلزم التخلص من الحقول المتعدية (Transitive) حتى تصبح الحقول معتمدة على حقل المفتاح فقط وليس على حقول أخرى وهذا هو مدخلنا إلى النوع الثالث من نماذج التطبيع.



شكل (٥ - ٧)

٥ - ١ - ٣ نموذج التطبيع الثالث (Third Normal Form)

نموذج التطبيع الثالث هو النموذج الذى يتخلص من العلاقات المتعدية (Transitive) بحيث تصبح الحقول معتمدة على حقل المفتاح فقط وليس على حقل آخر. ويتم ذلك عن طريق فصل الحقول المتعدية فى علاقة مستقلة. فمثلا العلاقة (Date - Type) يمكن تقسيمها إلى علاقتين مستقليتن وهما (Type)، (Date) أنظر شكل (٥ - ٨).

وبنموذج التطبيع الثالث تكون البيانات قد أصبحت فى صورة يسهل التعامل معها بواسطة الحاسب. وهذه الصورة لاتتغلب على مشكلة وجود حقل مثل (Service Type) كحقل فهرسى فى علاقة (Type) رغم كونه حقلا غير فهرسى فى علاقة (Date) لذلك فإن هناك تطبيعا رابعا وتطبيعا خامسا ولكن لامجال لدراستهما فى هذا الكتاب حيث أن نموذج التطبيع الثالث يعتبر كافيا فى معظم قواعد البيانات.

Service Date	Service Type	Service Place

Date — Type

Service Date	Service Type
11 / 89	Engine overhoul
3 / 90	Seat repair
12 / 91	Navigation system
2 / 91	Glass replacement
10 / 91	Engine Overhoul

Service Type	Service Place
Engine overhoul	H 03
Seat repair	H 07
Navigation system	H 10
Glass replacement	H 05
Engine Overhoul	H 03

شكل (٥ - ٨)

الفصل السادس

قواعد البيانات العلاقية

(Relational Database)



٦ - ١ مقدمـــة

يمكن تعريف النظم العلاقية (Relational Systems) بأنها النظم التى تتلقى البيانات من المستخدم على صورة جداول (Tables) ، ولاشيء غير الجداول ، ثم تستدعى عن طريق المشغل (Operator) على هيئة جداول جديدة تعتمد فى تصميمها وإخراجها على الجداول القديمة كما يتضح من الشكل (٦ - ١).

a) Given table

Student

Student #	Name	Class	Year
49	Aly	PC	1
50	Mohamed	PJ	2
115	Khaled	PA	1
90	Kareem	PB	4

b) Operators (Exampls)

Select Student No., Name, Class , Year
from Student where year = 1 ;

Student #	Name	Class	Year
49	Aly	PC	1
115	Khaled	PA	1

1. Row Subset

Select Name, Year
from Student ;

Name	Year
Aly	1
Mohamed	2
Khaled	1
Kareem	4

2. Column Subset

شكل (٦ - ١)

والشكل يوضح تراكيب البيانات والمعاملات (Operators) فى النظام العلاقى. ويلاحظ أن البيانات المعطاه تتكون من جدول واحد يسمى (Student) موضح فى جزء (a) من الشكل (٦ - ١) ويتم إستخراجها بطريقتين إما عن طريق مجموعات الصفوف (Row Subsets) أو مجموعات الأعمدة (Column Subsets) كما يتضح من

الجزء (b) من الشكل (٦ - ١).

ويلاحظ أيضا استخدام الأمر (SELECT) وهو أحد أوامر لغة البحث التركيبى (Structured Query Language) وتختصر (SQL) وهى إحدى لغات قواعد البيانات. وتستخدم النظم العلاقية فى تصميم قواعد البيانات وتسمى لذلك قواعد البيانات العلاقية (Relational Databases) وهذا النوع من قواعد البيانات يمثل الإتجاه الحديث لمعظم نظم قواعد البيانات ويعرف أيضا بنظام (SQL) أى أنه نظام علاقى (Relational) ويعتمد على لغة (SQL).

٦ - ٢ تعريف البيانات (Data Definition)

سوف نركز إهتمامنا فى هذا الجزء على الجمل الخاصة بتعريف البيانات (Data Definition) فى لغة (SQL) وبالتحديد الجزء العلاقى (Relational) فى هذه الجمل بمعنى أننا سوف نناقش ما يتعلق بالمستخدم مباشرة وليس مايخص المستوى الداخلى للنظام (Internal Level) مثل حجم الإسطوانة ومتطلبات عملية التخزين ... إلخ.

وكما سبق الإيضاح فإن نظم قواعد البيانات العلاقية تعتمد فى عمليات إدخال وإخراج المعلومات على الجداول والتى تسمى جداول القاعدة (Base Tables) شكل (٦ - ٢).

FLIGHT اسم الجدول

Flight No.	Date	Pilot ID	Aircraft No.
581	3 / 6	33461	1840T
452	2 / 28	41309	1849S
596	1 / 30	65348	1874J
428	1 / 24	13149	2174L
694	3 / 12	77685	1840T

شكل (٦ - ٢)

ويلاحظ أن الجدول مكون من الصف الأول وبه عنوان كل عمود من الأعمدة وبقية الصفوف تحتوى على وحدات البيانات بحيث تقابل كل وحدة بيانات عمودا معينا وكل عمود يجمع بدوره وحدات البيانات المتشابهة النوع.

ويلاحظ أيضا أن وحدات البيانات غير مرتبة ، ويعتبر ذلك من خصائص جداول القاعدة (Base Tables) ، لأن العلاقة (Relation) عبارة عن مجموعة من العمليات الرياضية التى تجرى صف بعد صف ولا تحتاج إلى ترتيب معين (Ordering). وفى حالة البحث خلال وحدات البيانات فى الجدول يمكن إخضاع مجموعة من الصفوف لترتيب معين.

وبعكس الصفوف التى لاتخضع لترتيب معين فإن الأعمدة يجب أن تكون مرتبة (Ordered) بحيث يكون العمود الأول ناحية الشمال وآخر عمود فى أقصى اليمين. فمثلا فى شكل (٦ - ٢) نجد أن العمود الأول هو رقم الرحلة (Flight No.) والأخير رقم الطائرة (Aircraft No.). ويأخذ كل جدول إسما معينا (Name) مثل إسم (FLIGHT) فى شكل (٦ - ٢) عن طريق أمر الإنشاء (CREATE) وفيما يلى بعض الأوامر التى تستخدم فى تعريف البيانات.

أ - إنشاء جدول (CREATE TABLE)

الشكل العام لجملة إنشاء جدول كالآتى :

```
CREAT TABLE base - table - name
    ( Column - definition [ , Column - definition ] ... ) ;
```

والمثال الآتى يوضح استخدام هذه الجملة فى إنشاء الملف (Flight) الموضح فى شكل (٦ - ٢).

```
CREATE TABLE  FILGHT
FLIGHT #            SMALLINT  NOT  NULL ,
DATE                DATE
PILOT ID            INTEGER,
AIRCRAFT #          CHAR ( 5 )
```

وعند إدخال هذا الأمر إلى الحاسب يقوم النظام ببناء جدول القاعدة المسمى (FLIGHT) ويكون الجدول فارغا إلا من السطر الذى يحتوى على عناوين الأعمدة مثل رقم الرحلة (Flight No.) ، التاريخ ، ... الخ. ثم يتم إدخال البيانات عن

طريق أمر الحشر (INSERT). ويلاحظ أن نوع البيانات يذكر بجانب عنوان كل عمود. وفى نظام (SQL) يوجد الأنواع الآتية من البيانات :

الرقم الصحيح (INTEGER) ، الرقم الصحيح الصغير (SMALLINT) ، الرقم العشرى (DECIMAL) ، والرقم ذو النقطة المتحركة (FLOATING) والحروف (CHAR (n)) حيث (n) هو عدد الحروف فى الكلمة ، والحروف متغيرة الطول (VAR CHAR (n)) والتاريخ (DATE) والقيمة غير المحددة .(NULL)

ب - تعديل الجدول (ALTER TABLE)

بعد إنشاء الجدول يمكن إضافة أى عمود من اليمين باستخدام أمر التعديل (Alter Table) كالآتى :

```
ALTER TABLE base - table - name
            ADD column - name  data  type :
```

والمثال الآتى يوضح كيفية إضافة عمود إلى الجدول (Flight) الموضح فى الشكل (٦ - ٢)

```
ALTER TABLE FLIGHT
            ADD  PILOT NAME CHAR ( 20 ) ;
```

ويلاحظ من الجدول أن العمود إسم الطيار (Pilot Name) قد تم تحديد نوع البيانات فيه (CHAR (20)) وهو حرفى بعدد ٢٠ حرف والشكل (٦ - ٣) يوضح الجدول فى شكل (٦ - ٢) بعد الإضافة.

حـ - إلغـاء جـدول (Drop Table)

يستعمل هذا الأمر لإلغاء جدول موجود فى النظام فى أى وقت والشكل العام له كالآتى :

```
DROP TABLE  base - table - name ;
```

FLIGHT

Flight No.	Date	Pilot ID	Aircraft No.	Pilot Name
581	3 / 6	33461	1840T	Mohamed
......	......	....	....	....
......	......	....	....	....

شكل (٦ - ٣)

ويمكن باستخدام هذا الأمر إلغاء الجدول (Flight) كالآتى :

```
DROP TABLE FILGHT ;
```

د - الفهارس (Indexes)

أمر إنشاء الفهارس يكون كالآتى :

```
CREAT { UNIQUE } INDEX index - name
ON base - Table - name ( Column - name - { order }
{ Column - name {order } }........)
```

والترتيب (Order) يكون إما تصاعديا (Ascending) أو تنازليا (Descending) وفى حالة عدم ذكر الترتيب يتم فرضه تصاعديا مباشرة. على سبيل المثال لإنشاء فهرس الجدول (FLIGHT) يكتب الأمر كالآتى :

```
CREATE UNIQUE INDEX XFLIGHT ON ( FLIGHT NO)
```

وكلمة منفرد (Unique) تعنى أنه لايوجد سجلان فى الفهرس يأخذ كل منهما نفس القيمة لحقل الفهرس (Index Field) ويمكن أيضا إلغاء الفهرس

باستخدام الأمر (DROP INDEX) كالآتى :

```
DROP INDEX  XFLIGHT ;
```

٦ - ٣ تشغيل البيانات (Data Manipulation)

بعد تعريف البيانات تأتى عمليات التشغيل (Mainpulation). ويسمح النظام بتشغيل البيانات باستخدام مجموعة من الأوامر مثل أمر الإختيار (SELECT) وأمر التحديث (UPDATE) ، وأمر الحذف (DELETE) وأمر الحشر (INSERT) وسنتناول فى هذا الجزء خواص هذه الأوامر وكيفية استخدامها فى عمليات التشغيل.

٦ - ٣ - ١ البحـــث (Query)

سنبدأ بمثال بسيط وهو البحث عن أرقام الرحلات التى ستقوم بها الطائرة رقم (1840 T) وتاريخها.

```
SELECT FLIGHT No. , DATE
FROM  FLIGHT
WHERE AIRCRAFT No. = " 1840 T "
```

ويكون ناتج عملية البحث كالآتى :

Result :

FLIGHT No.	DATE
581	3/6
694	3/12

ويلاحظ أن أمر الإختيار (SELECT) يقوم بإختيار الحقول من جدول معين (FLIGHT) والتى تحقق شرطا معينا.

وأمر الإختيار (SELECT) له صور كثيرة نستعرضها باستخدام الجداول الموضحة بالشكل (٦ - ٤).

S — جدول (١) المورد (Supplier)

Supplier #	SNAME	CITY	ACC.
S1	Loutfy	Cario	2000
S2	Khaled	Alex	1500
S3	Aly	Cario	1000

P — جدول (٢) الجزء ومكان تخزينه (P)

Part #	PNAME	WEIGHT	CITY
P1	Nut	12	Cario
P2	Bolt	17	Cario
P3	Screw	17	Alex
P4	Screw	14	Cario

SP

Supp #	Part #	Qty
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S2	P1	300
S2	P2	400
S3	P2	200

جدول (٣)
المورد - الجزء - الكمية (SP)

شكل (٦ - ٤)

١ - الإسترجاع البسيط (Simple Retrieval)

لإيجاد رقم الجزء (Part #) لكل الأجزاء التى تم توريدها (SP) نكتب :

```
SELECT Part #
From SP ;
```

Result :

Part #
P1
P2
P3
P4
P1
P2
P2

ويلاحظ تكرار رقم الجزء فى هذه النتائج. ولغة (SQL) لاتحذف التكرار فى النتائج إلا إذا طلب المستخدم ذلك باستخدام أمر التمييز (DISTINCT) كالآتى :

```
SELECT DISTINCT part #
FROM SP ;
```

Result :

Part #
P1
P2
P3
P4

ب - إسترجـاع التعبيـر (Retrieval of Expression)

لتحديد رقم الجزء ووزنه بالجرام مع ملاحظة ان وزن الجزء (WEIGHT) فى الجدول (P) معطى بالباوند ، نكتب الآتى

```
SELECT  P.part # ,  ' Weight in grams = ', P. WEIGHT * 454
FROM P ;
```

Part #		
P1	Weight in grams	5448
P2	Weight in grams	7718
P3	Weight in grams	7718
P4	Weight in grams	6356

جـ - الإسترجاع التفصيلى

لإيجاد تفاصيل كاملة عن كل الموردين (Suppliers) :

```
SELECT S. *
From S ;
```

والناتج هنا هو نسخة كاملة من جدول الموردين (S) كالموضح فى شكل (٦ - ٤). والعلامة (*) تعنى كل الحقول فى الجدول (S).

د - الإسترجاع المـــاهر (Qualified Retrieval)

لإيجاد كل الموردين الموجودين فى القاهرة والذين لهم حساب(Account) يزيد عن ١٠٠٠ جنيه يكون الأمر كالآتى :

```
SELECT SUPPLIER #
FROM  S
WHERE CITY = ' Cairo '
AND ACC > 1000
```

Reseult :

Supplier #
S 1

وتأتى المعاملات المنطقية مثل (... = > و = < و > و < و =) دائما مع (WHERE) لاستخدامها فى عمليات المقارنة.

هـ - الإسترجاع بالترتيب (Retrieval With Ordering)

لإيجاد أرقام الموردين فى القاهرة وحساباتهم بترتيب تنازلى نكتب الأمر الآتى :

```
SELECT Supplier # , ACC
FROM S
Where CITY = ' Cairo '
ORDER BY ACC DESC :
```

Result :

Supplier #	ACC
S1	2000
S3	1000

٦ - ٣ - ٢ الإسترجاع بالربط (Join Queries)

من أهم خصائص النظم الفرعية العلاقية هى قدرتها على الربط (Join) بين جدولين أو أكثر فى جدول واحد. وهناك حالات متعددة للإسترجاع بالربط نذكر منها الآتى :

أ - الربط المتساوى البسيط (Simple-Equi Join)

لإيجاد كل الموردين والمعلومات عن الأجزاء التى يتم توريدها من نفس المدينة يكون الأمر كالآتى :

```
SELECT S.* , P.*
FROM S,P
WHERE S. CITY = P. CITY
```

Result :

Supplier #	S.NAM	S.CITY	ACC	Part #	P NAME	WEIGHT	P.CITY
S1	Loutfy	Cairo	2000	P1	Nut	12	Cairo
S1	Loutfy	Cairo	2000	P2	Bolt	17	Cairo
S1	Loutfy	Cairo	2000	P4	Screw	14	Cairo
S2	Khaled	Alex	1500	P3	Screw	17	Alex
S3	Aly	Cairo	1000	P4	Screw	14	Cairo

ويلاحظ أن جملة حيث (WHERE) اقترن فيها إسم الحقل (City) بإسم الجدول لتحديد عملية البحث. ويتم ربط بيانات الجدول (S) بالجدول (P) عن طريق تساوى قيم حقل المدينة (City) فى الجدولين.

ب - الربط الزائد (Greater - Than Join)

لإيجاد كل المعلومات عن الموردين والأجزاء بحيث تكون مدينة المورد (Supplier City) تلى مدينة الجزء (Part City) فى ترتيب حرفى (Alphabetical Order) نكتب التالى :

```
SELECT S.* , P.*
FROM S , P
WHERE S.CITY > P. CITY ;
```

Supplier #	S.NAM	S.CITY	ACC	Part #	P NAME	WEIGHT	P.CITY
S1	Loutfy	Cairo	2000	P3	Screw	17	Alex

ولإسترجاع حقل معين بالربط وليس الجدول كله نكتب :

```
SELECT S. Supplier # , P. Part #
FROM S , P
Where S.CITY = P.CITY ;
```

Result :

Suppleir #	Part #
S1	P1
S1	P2
S1	P4
S2	P3
S3	P4

حـ - ربط ثلاثة جداول (Join of Three Tables)

لإيجاد إسم مدينة المورد (Supplier City) وإسم المدينة المخزن فيها الأجزاء (Part City) الخاصة بهذا المورد نجد أننا فى حاجة إلى ربط الجداول (١ ، ٢ ، ٣) فى شكل (٦ - ٤) وذلك باستعمال الأوامر التالية :

```
SELECT    DISTINCT  S.CITY ,  P. CITY
FROM      S , SP , P
WHERE     S.Supplier # = SP. Supplier #
AND       S.Part  #  =  P. Part #
```

Result :

S.CITY	P.CITY
Cairo	Cairo
Cairo	Alex
Alex	Cairo

نلاحظ أن أمر التمييز (Distinct) منع تكرار إزدواج البيانات المتشابهة فى حقل (P. City) لنفس (S. City).

د - ربط الجدول بنفسه (Join a Table with Itself)

يمكن إيجاد أسماء الموردين من نفس البلد من جدول الموردين (S) كالآتى :

```
SELECT FIRST . SUPPLIER # , SECOND . Supplier #
FROM  S  FIRST, S SECOND
WHERE FIRST.CITY = SECOND.CITY
AND FIRST . Supplier # <  SECOND Supplier #
```

Result :

Supplier #	Supplier #
Cairo	Cairo

ويلاحظ هنا ربط الجدول بنفسه عن طريق المدينة (City) ولهذا يظهر مرتين فى جملة (From) لذلك تتم إضافة كلمة (FIRST , SECOND) إلى إسم الجدول لتسهيل عمل جملة الشرط المكونة من (SELECT - WHERE).

٦ - ٣ - ٣ الوظائف المبنية (Built - IN Functions)

مما سبق شرحه نلاحظ ان أمر الإختيار (SELECT) من الأوامر المؤثرة والواسعة الإستخدام فى عملية تشغيل البيانات ومع هذا هناك بعض المشاكل التى تواجه المستخدم عند استعمال هذا الأمر. من هذه المشاكل مثلا أن المستخدم لايستطيع معرفة العدد الكمى للموردين ولهذا يتم الإستعانة ببعض الوظائف المبنية فى النظام (Built - in Functions) مثل العدد (Count) والجمع (SUM) والمتوسط (AVG) والأكبر (MAX) والأصغر (MIN). ويتم تطبيق هذه الوظائف على مجموعة من وحدات البيانات الموجودة فى عمود معين فى جدول ما.

وفيما يلى بعض الأمثلة ونتائجها والتى توضح إستعمال هذه الوظائف.

```
1 - SELECT COUNT ( * )
      FROM  S
```

Result :

Supplier #
-
-
-
3

```
2 - SELECT SUM ( * )
FROM SP
WHERE Part # = " P1 "
Result :
```

Part #
-
-
-
-
4

٦ - ٣ - ٤ عمليات التحديث (Update Operations)

سنوضح بالأمثلة أوامر تحديث مجموعة من وحدات البيانات فى قاعدة بيانات علاقية

أ - أمر التحديث (Update)

```
UPDATE      Table
SET              Field = Expression
            { , Field = Expression } .......
{ WHERE Predicate } ;
```

وهذا الأمر يعنى أن كل وحدات البيانات التى تحقق الشرط (Predicate) يتم تحديثها وفقا للتعبير (Expression) الموجود أمام الحقل. ومثال لذلك.

```
UPDATE  P
SET  WEIGHT  = WEIGHT + 5 ,
WHERE  Part # = " P2 " ;
```

أى أن عملية التحديث ستتم فى الجدول (P) وأن كل الأجزاء (P2) يتم زيادة وزنها بخمسة.

ب - أمـر الحـذف (DELETE)

يأخذ أمر الحذف الصورة الآتية :

```
DELETE
FROM  table
{ WHERE Predicate }
```

ومثال ذلك

```
DELETE
FROM   S
WHERE CITY = ' Alex '
```

حـ - أمـر الحشـر (INSERT)

يأخذ أمر الحشر الصورة الآتية :

```
INSERT
INTO  table { ( Field { , Field } .... ) }
VALUES ( Constant { , Constant }..... ) ;
```

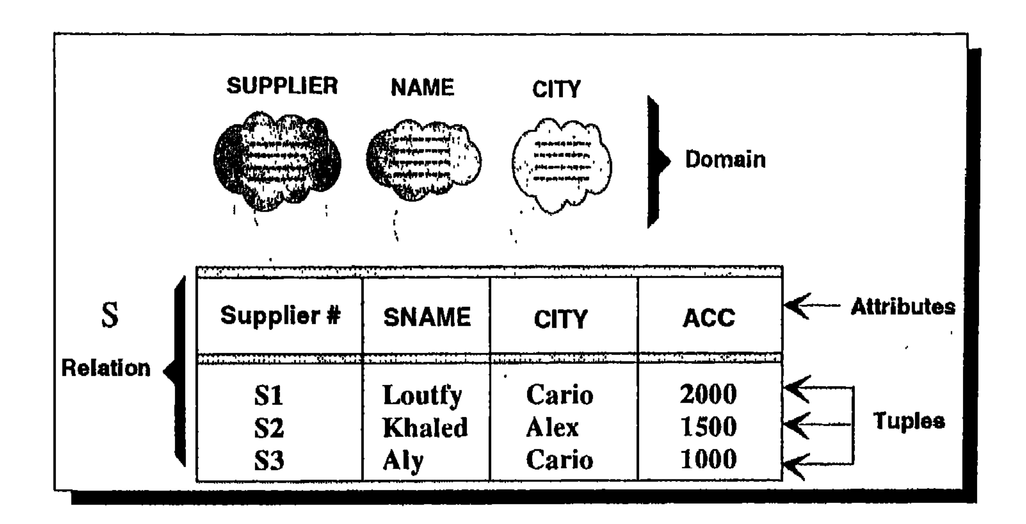
ومثال ذلك

```
INSERT
INTO        P  ( Part # , CITY , WEIGHT )
VALUES       ( ' P5 '  , ' Port  Said ' , 24 )
```

ويلاحظ هنا أن جزءا جديدا قد تم إضافته للجدول (P) ومكان تخزينه بورسعيد ووزنه ٢٤ رطل ولكن إسمه غير محدد (NULL).

٦ - ٤ العلاقات (Relations)

تحتاج العلاقات فى تعريفها إلى ما يسمى النطاق أو الحيز (Domain) وهو الوعاء المحتوى على البيانات أنظر شكل (٦ - ٥). فمثلا نطاق المدينة (CITY) يحوى مجموعة المدن (Set) ومدينة القاهرة (Cairo) تعتبر مجموعة فرعية (Supset) من مجموعة المدن.



Supplier #	SNAME	CITY	ACC
S1	Loutfy	Cario	2000
S2	Khaled	Alex	1500
S3	Aly	Cario	1000

شكل (٦ - ٥)

وكل نطاق له إسم يميزه (Attribute) ويكون موجودا فى الصف الأول من أى جدول. ولأن الجدول بمحتوياته يصف علاقة معينة فإن كلمة علاقة (Relation) هنا تعنى عند إستخدامها جدولا محددا.

فمثلا الجدول فى شكل (٦ - ٥) والمسمى (S) يمكن تسمته بعلاقة المورد (Supplier Relation). ويمكن تقسيم العلاقة (Relation) أفقيا إلى قسمين وهما العناوين (Heading) والجسم (Body).

وتحتوى العناوين على مجموعة محددة من الأسماء المميزة يتم كتابتها لكل عمود أما الجسم فيحتوى على مجموعة محددة من وحدات البيانات تتغير مع الوقت

(Time - Varying) عند تحديث العلاقة. فمثلا العلاقة الموضحة فى شكل (٦ - ٥) تحتوى على العناوين الآتية رقم المورد (Supplier #) وإسمه (SNAME) ومدينته (City) والحساب (Acc) أما الجسم فيحتوى على ثلاثة صفوف (Tuples).

ولكل علاقة درجة (Degree) معينة تحدد بعدد الأعمدة فى هذه العلاقة فمثلا العلاقة (S) من الدرجة الرابعة والعلاقة (P) من الدرجة الرابعة. والعلاقة (SP) من الدرجة الثالثة أنظر شكل (٦ - ٤). وأيضا لكل علاقة عدد من الصفوف (Cardinality) ويلاحظ أن عدد الأعمدة فى أى علاقة ثابت بينما عدد الصفوف يتغير حسب عمليات التحديث كما سبق الإيضاح ومن الخصائص الرئيسية التى تميز العلاقات (Relations) الآتى :

أ - لاتسمح بتكرار الصفوف (Tuples).
ب - الصفوف لاتخضع لترتيب معين (Unordered).
حـ - الاعمدة لاتخضع لترتيب معين.

وهذا التعريف بالعلاقات يؤدى إلى بنا إلى تعريف أدق وأشمل لقواعد البيانات العلاقية. حيث يمكن تعريفها بأنها مجموعة من العلاقات المتغيرة زمنيا والتى يمكن تطبيعها وترتيبها بواسطة المستخدم وفقا لنظام معين.

وهناك تشابه بين العلاقة والملف (File) حيث يماثل الصف (Tuple) والعمود (Attribute) فى العلاقة السجل (Record) والحقل (Field) فى الملف من حيث النوع وليس الحدوث (Occurrence). وهذا التشابه أدى إلى تسمية العلاقة بالملف العلاقى (Relational File) لتمييزه عن الملفات النمطية العادية. وهذا الملف العلاقى يحتوى على سجل خاص بكيان واحد مثل سجل الموردين (Suppliers) أو سجل الجزء (Part) وكل حقل فى الملف له قيمة واحدة مثل حقل (S3) فى رقم المورد (Supplier#). والسجلات فى هذا الملف تحتوى على حقل مميز واحد (Unique Identifier) يسمى المفتاح الرئيسى (Primary Key).

٦ - ٥ المفاتيح الرئيسية (Primary Keys)

المفاتيح الرئيسية هى عبارة عن نموذج خاص من تركيب عام يسمى المفاتيح المنتخبة (Candidate Keys). والمفتاح الرئيسى هو عبارة عن مميز واحد(Unique Indetifier) فمثلا المفتاح الرئيسى فى علاقة المورد (Supplier , S) هو رقم المورد (Supplier #).

ومن الصفات الخاصة للمفاتيح الرئيسية صفة الفردية (Uniquesness) أى عدم التكرارية فمثلا لايمكن أن يكون لإثنين من الموردين نفس الرقم.

٦ - ٦ المفاتيح الثانوية (Secondary Keys)

المفاتيح الثانوية عبارة عن عمود مميز (Attribute) أو خليط من الأعمدة المميزة فى علاقة ما (R2) بحيث تماثل قيم وحدات البيانات فى هذا العمود القيم المناظرة لها فى المفتاح الرئيسى لعلاقة أخرى (R 1) وليس بالضرورة أن تكون العلاقتان (R1 , R2) مختلفتين تماما. فمثلا الأعمدة رقم المورد (# Supplier) ورقم القطعة (# Part) فى الجدول (SP) شكل (٦ - ٤) يمثل كل منهما مفتاحا ثانويا للعلاقة (SP) حيث أن قيمة كل منهما يجب أن تماثل القيم المناظرة للمفاتيح الرئيسية فى العلاقة (Supplier) والعلاقة (Part) وهما (# S , # P) بمعنى أن العمود (Supplier) فى الجدول (SP) لايمكن أن يحتوى على رقم المورد (S4) لأنه غير موجود فى المفتاح الرئيسى (# Supplier) فى العلاقة (S). أيضا لايمكن أن يحتوى على رقم القطعة (P6) لأنها غير موجودة فى العلاقة (P).

والتماثل (Match) فى العلاقة بين المفاتيح الرئيسية والثانوية يعتبر بمثابة الرباط الذى يحافظ على وحدة قاعدة البيانات العلاقية ومن أمثلة ذلك أن أى قيمة من قيم المفتاح الثانوى لايمكن أن تكون خالية (Null) فى نفس الوقت الذى تكون فيه كل قيم المفتاح الرئيسى لها قيمة.

٦ - ٧ التأمين (Security)

تتعرض وحدات البيانات فى قاعدة معينة إلى الإستيلاء أو التدمير. والتأمين هنا يعنى السماح فقط للأشخاص المسئولون (Authorized) وليس غيرهم بتشغيل البيانات داخل القاعدة ضمانا لها من التدمير أو التلف. ويتم ذلك عن طريق وضع قيود معينة (Constraints) على نظام التشغيل ويقوم بصياغتها المسئولون عن إدارة قاعدة البيانات (DB Adminstration) وبلغة معروفة ثم تحفظ فى كتالوج أو قاموس معين بحيث يسهل الرجوع إليها عند الحاجة. وهناك إعتبارات عديدة تتعلق بمشكلة التأمين منها :

١ - إعتبارات شرعية (Legal) وإجتماعية (Social) فمثلا هل الشخص الذى يطلب رقم حساب الضمان لأحد العملاء. مصرح له بذلك أم لا؟
٢ - أعتبارات الحماية الطبيعية ، بمعنى هل يجب إحكام غلق حجرة الحاسبات أو حراستها.

٣ - إعتبارات مشاكل التشغيل ، على سبيل المثال هل يستخدم نظام كلمة السر (Password) وكيف يتم المحافظة على سرية الكلمات.

٤ - إعتبارات حماية المكونات المادية (Hardware) بمعنى هل تتمتع مثلا وحدة المعالجة المركزية (CPU) بأى حماية مثل مفاتيح حماية المخزون (Storage Protection Keys).

٥ - إعتبارات حماية نظام التشغيل مثل هل يقوم نظام التشغيل بمسح محتويات المخازن والملفات عند الإنتهاء منها.

ويتباين حجم وحدات البيانات المراد تأمينها بين صف واحد أو عمود واحد داخل جدول إلى مجموعات عديدة من الجداول.

وتتمتع بعض نظم قواعد البيانات مثل (SQL) بما يسمى بآلية المشاهدة (View Mechancism) بمعنى القدرة على إخفاء البيانات الهامة من جدول معين عند عرضه على مستخدم غير مصرح له (Unauthorized) وإظهارها للمستخدم المصرح له عن طريق إختبار متطلبات التشغيل (Access Request) على سبيل المثال إذا كانت القيود على نظام التشغيل فى بنك ما لاتسمح إلا لمدير البنك بالإطلاع على أرقام الإئتمان للعملاء فإن أى موظف لا يستطيع معرفة رقم الإئتمان لعميل ما (من الجدول الخاص به) لأنه لا يحقق متطلبات نظام التشغيل فى نفس الوقت الذى يستطيع فيه هذا الموظف معرفة رقم الحساب لأن نظام التشغيل يسمح بذلك كما يتضح من الشكل (٦ - ٦).

ويستخدم أمر (VIEW) لأغراض الحماية. فمثلا الجدول المسمى (C) الموجود فى شكل (٦ - ٦ - أ) هو نتيجة الأمر الآتى :

```
CREATE VIEW CUSTOMER _ CREDIT
   AS SELECT CUSTOMER # , C NAME , CADRESS,
          C ACCOUNT # , CREDIT #
   FROM C ;
```

أ - جدول العملاء (فى حالة طلب المدير)

Customer No.	CNAME	CADDRESS	CACCOUNT No.	Credit No.
12718	Aly S.	Roxy	1790	217

ب - جدول العملاء (فى حالة طلب الموظف)

Customer No.	CNAME	CADDRESS	CACCOUNT No.	Credit No.
12718	Aly S.	Roxy	1790	

شكل (٦ - ٦)

أما الجدول (٦ - ٦ - ب) فهو نتيجة الأمر الآتى :

```
CREATE VIEW CUSTOMER _ ACCOUNT
    AS SELECT CUSTOMER # , ...... , ACCOUNT #
        FROM C ;
```

ويعتبر الأمر (VIEW) والذى يحقق آلية المشاهدة من طرق الحماية غير المكلفة لأنه أحد أوامر النظام ولكن من الأنواع التى تستعمل اثناء التشغيل وليس قبله.

وأى نظام للحماية لايمكن أن يحقق الكمال (Perfect System) لذلك يقوم النظام بإنشاء ملف خاص أو قاعدة بيانات ويحتفظ آليا ببيانات كاملة عن نوع عملية التشغيل ، رقم الكمبيوتر أو الوسيلة التى تم إجراء عملية التشغيل منها ، وإسم المستخدم ، تاريخ ووقت التشغيل ، قاعدة البيانات والجداول والسجلات والحقول التى خضعت لعملية التشغيل ، القيم السابقة للحقول وأخيرا القيم الجديدة للحقول.

٦ - ٨ التكامل (Integrity)

تكامل البيانات يعنى صحة ودقة البيانات داخل قاعدة البيانات وهناك قيود يفرضها النظام تسمى قيود التكامل (Integrity Constraints) والآتى مثال بسيط لهذه القيود :

```
S. ACCOUNT > 0
```

بمعنى أن حساب العملاء فى بنك ما يجب أن يكون موجبا (Positive) حتى يستطيع سحب مبلغ معين. وفى حالة محاولة السحب مع عدم تحقق هذا الشرط فإن عملية التشغيل كلها تتوقف لمخالفتها للقيد المذكور.

ومن القيود التى تفرض على النظام لضمان تكامل البيانات تلك التى تفرض على نطاق من البيانات وتسمى قيود النطاق (Domian Constraints) فمثلا نطاق الأسماء (Name) يجب أن يحتوى على أسماء فقط وليس أرقام وذلك لتحقيق شرط توافق نوعية البيانات مع نوعية الحقول. وأيضا يعتبر توافق المفاتيح الثانوية مع المفاتيح الرئيسية أحد القيود لتحقيق عملية التكامل.

وتستخدم بعض الأوامر مثل الأمر (CREATE CONSTRAINT) لتحقيق مبدأ تكامل البيانات كالآتى :

```
CREATE CONTSTRAINT C1
CHECK  S. ACCOUNT > 0
EISE reject  Operation ;
```

ويلاحظ من هذه الأمر أن لكل قيد (Constraint) إسما مثل (C1) وفى حالة تحقق حدوث القيد فإن النظام سوف يرفض العملية ويوقف تنفيذها.

ويمكن إستعمال أمر إسقاط القيد (DROP CONSTRAINT) فى حالة عدم الرغبة فى إستعماله كالآتى :

```
DROP CONSTRAINT C1 ;
```

وتعانى معظم نظم قواعد البيانات فى الوقت الحالى من عدم توفير طرق تكامل البيانات بطريقة كاملة بالمقارنة بطرق التأمين وهناك محاولات عديدة للتطوير فى هذا الإتجاه.

الفصل السابع

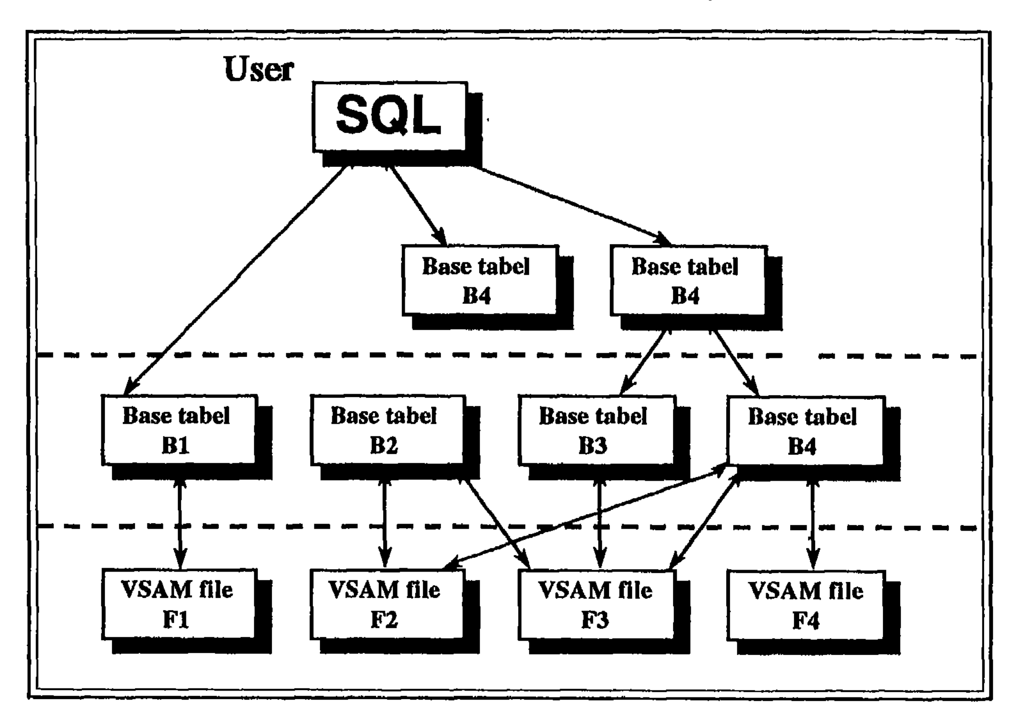
لغـــة (SQL)

كنموذج لقواعد البيانات العلاقية



كما ذكرنا من قبل فإن معظم نظم قواعد البيانات فى الوقت الحاضر من النوع العلاقى (Relational) بالإضافة إلى استخدام معظم هذه النظم للغة (SQL) لذلك أطلق عليها نظم (SQL) وكلمة (SQL) أصلها هو (SEQUEL) وهو الإسم الذى أطلقه معمل التطوير والأبحاث بشركة (IBM) على اللغة الجديدة التى تم تطويرها لاستخدامها فى تصميم نظم إدارة قواعد البيانات. وكلمة (SQL) إختصار للكلمة (Structured Query Language) أى لغة البحث المركبة لأنها لغة بحث وفى نفس الوقت لغة لبرمجة قواعد البيانات.

وتستخدم لغة (SQL) لتمثيل قواعد البيانات العلاقية وتتيح للمستخدم إمكانيات كبيرة ومرونة فائقة لتشغيل البيانات. والمستخدم هنا نوعان إما مستخدم لنهاية طرفية (End-User) أو مبرمج تطبيقات (Application Programmer).



شكل (٧ - ١)

والشكل (٧ - ١) يوضح استخدام لغة (SQL) بواسطة المستخدم. ويلاحظ أن المستخدمين على اختلاف أنواعهم يقومون بتشغيل نفس البيانات فى نفس الوقت حيث يمكن التحكم فى عمليات التشغيل بحيث يتم حماية كل مستخدم من التأثير على

المستخدم الآخر. بمعنى أن أى عملية تحديث للبيانات يقوم بها أحد المستخدمين لاتؤثر على نتائج مستخدم آخر يعتمد على نفس البيانات ويتيح النظام التعامل مع نوعين من الجداول : جداول القاعدة (Base Tables) وهى جداول حقيقية والمناظر أو الصور (Views) وهى جداول مخلقة من الجداول الأصلية وبصور مختلفة

وتتيح لغة (SQL) للمستخدم التعامل مع أكثر من جدول لقاعدة (Base Table) أو صورة مثل (VSAM File) فى نفس الوقت.

وتحتوى لغة (SQL) على جزئين رئيسين ، جزء خاص بتعريف البيانات (Data Definition Langauge) أو (DDL) وجزء خاص بتشغيل البيانات (Data Manipulation Langauge) أو (DML) وسوف نستعرض فى الجزء التالى المكونات الرئيسية لنظام (SQL).

٧ - ١ المكونـات الرئيسيـة

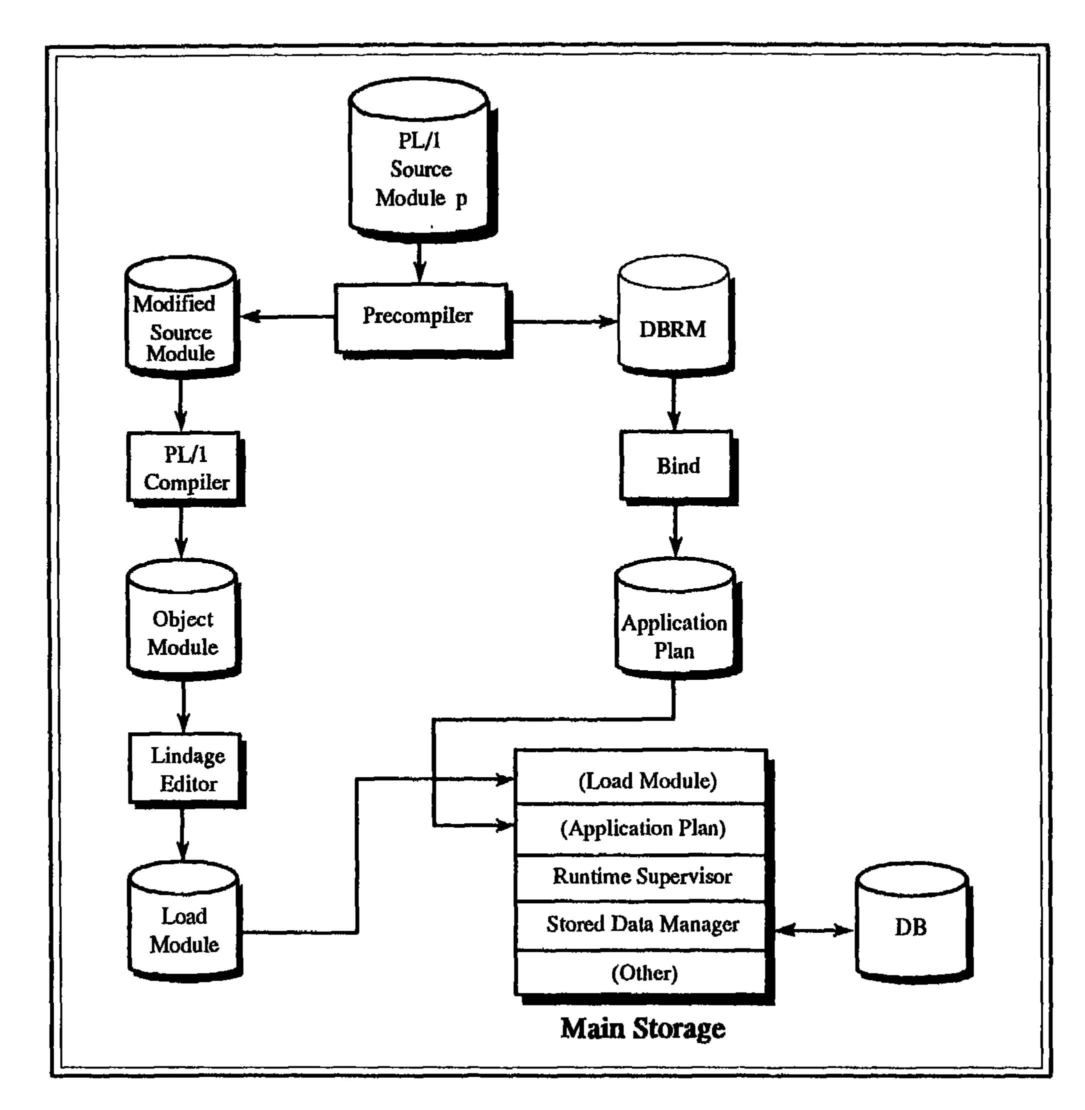
يمكن تقسيم المكونات الرئيسية إلى أربعة أجزاء وذلك من وجهة نظر المستخدم وهى جزء ما قبل برنامج الترجمة (Precompiler) ، جزء الربط (Bind) ، جزء مراقب زمن التشغيل (Runtime Supervisor) وأخيرا مدير البيانات المخزنة (Stored Data Manager) انظر شكل (٧ - ٢).

٧ - ١ - ١ جزء ما قبل برنامج الترجمة (Precompiler)

هذا الجزء عبارة عن معالج (Processor) لبرامج التطبيق (Application Programms) التى تحتوى على جمل لغة (SQL). ويقوم بتجميع هذه الجمل داخل نموذج قاعدة بيانات جاهز للطلب(Database Request Module) أو (DBRM). ثم إستبدالها فى البرنامج الأصلى بمجموعة نداءات (CALLS) للجزء المراقب لزمن التشغيل.

٧ - ١ - ٢ جـزء الربـط (Bind)

يؤدى هذا الجزء دور الترجمة لواحد أو أكثر من (DBRM) للحصول على كود الآلة (Machine Code) الخاص بها بما فيها النداءات لمدير البيانات المخزونة (Stored Data Manager).



شكل (٧ - ٢)

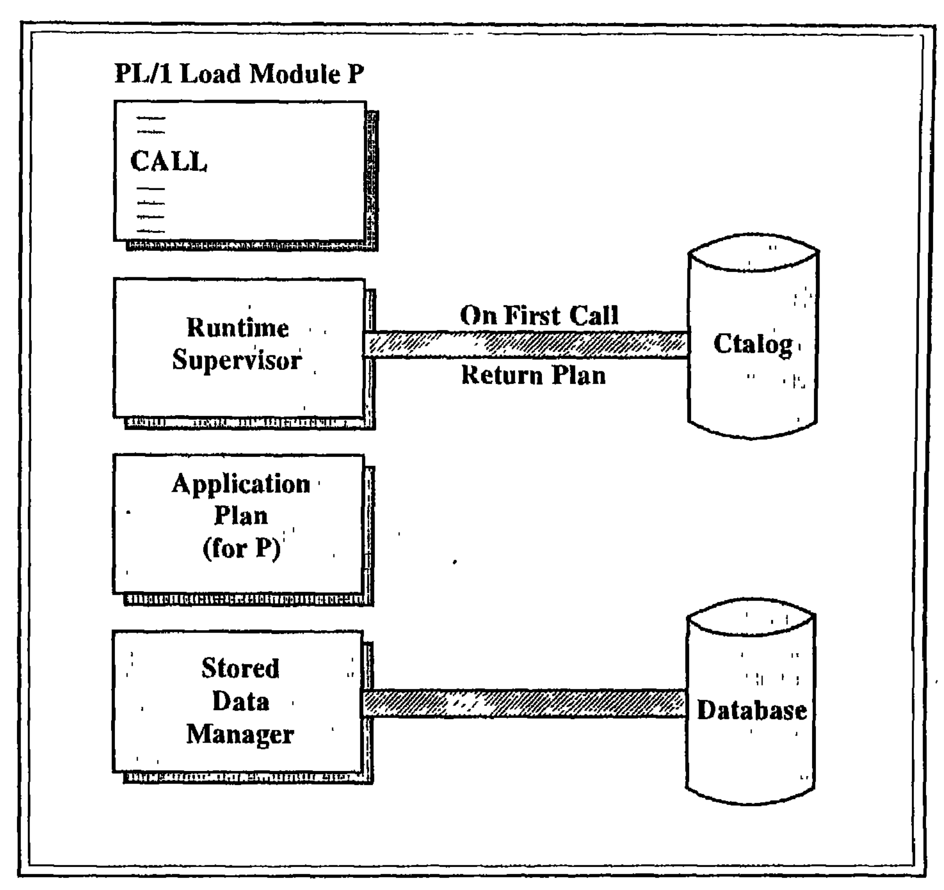
٧ - ١ - ٣ جزء مراقب زمن التشغيل (Runtime Supervisor)

يقوم هذا الجزء بمراقبة تنفيذ برامج (SQL) أثناء التنفيذ. فمثلا عند طلب بعض عمليات قواعد البيانات من قبل برنامج معين فإن وحدة التحكم(Controller) توجه هذا الطلب أولا إلى مراقب زمن التشغيل وذلك عن طريق النداءات التى يضيفها جزء ماقبل برنامج الترجمة (Precompiler). ثم من مراقب زمن التشغيل

إلى خطة التطبيق (Application Plan) والتى توجه مدير البيانات المخزونة لتنفيذ العملية المطلوبة.

٧ - ١ - ٤ مدير البيانات المخزونة (Stored Data Manager)

يقوم مدير البيانات المخزونة بإدارة قاعدة البيانات الفعلية بالإضافة الى تخزين واسترجاع السجلات حسب الحاجة انظر شكل (٧ - ٣).



شكل (٧ - ٣)

٧ - ٢ العمليات التى لاتستخدم المؤشر (Operations Not Involving Cursors)

العمليات التى لاتستخدم المؤشر يمكن تصنيفها كالآتى :

- الاختيار الفردى (Single Select)
- التحديث (Update)
- الحذف (Delete)
- الحشر (Insert)

وفيما يلى بعض الأمثلة التى توضح هذه العمليات.

٧ - ٢ - ١ الاختيـار الفـردى

ويعنى أن أمر الاختيار (Select) يقوم باسترجاع جدول مكون من صف واحد. فمثلا لايجاد مدينة (City) وحساب (Account) خاص بمورد معين تم تحديد رقمه باستخدام الأمر (GIVEN S #) نكتب الآتى :

```
EXEC  SQL  SELECT ACC, CITY
           INTO : RANK ,  CITY
           FROM   S
           WHERE S# = : GIVEN S # ;
```

وفى هذا المثال نجد أن أى سجل فى الجدول (Supplier) يحقق الشرط فى جملة (WHERE) ، يؤدى الى تمرير القيم (ACC , CITY) إلى المتغيرات (CITY , RANK) مع إعطاء (SQLCODE) القيمة صفر. أما إذا لم يحقق أى سجل الشرط فى جملة (WHERE) فإن قيمة (SQLCODE) تكون (100+). وفى حالة وجود أكثر من سجل يحقق الشرط فى جملة (WHERE) فإن قيمة (SQLCODE) تكون سالبة (Negative) مما يسبب خطأ فى البرنامج.

٧ - ٢ - ٢ التحـــديث

فى حالة زيادة رصيد كل الموردين فى مدينة القاهرة بقيمة معينة باستخدام المتغير (Raise) نكتب الآتى :

```
EXEC SQL UPDATE SUPPLIER
           SET ACC = ACC + : RAISE
           WHERE CITY = ' CAIRO ' ;
```

وأيضا إذا لم يحقق أى سجل الشرط فى جملة(WHERE) فإن (SQLCODE) سيأخذ القيمة (100+).

٧ - ٢ - ٣ الحـــــذف

لحذف كل شحنة (Shipment) لأى مورد من البلد الذى تم تحديده عن طريق المتغير (CITY) فإن الأمر يكون كالآتى :

```
EXEC SQL DELETE
        FROM SP
        WHERE : CITY =
                (SELECT CITY
                FROM SUPPLIER
                WHERE Supplier S# = SP.S#) ;
```

٧ - ٢ - ٤ الحشـــر

لحشر جزء جديد (New Part) فى الجدول (P) ومايخصه من رقم (Part Number) وإسم (Name) ووزن (Weight) عن طريق المتغيرات (PNO,PNAME,PWT) على التتابع يتم كتابة الآتى :

```
EXEC SQL INSERT
        INTO P( PART# , PNAME , WEIGHT)
        VALUES ( : PNO , : PNAME , : PWT)
```

٧ - ٣ العمليات التى تستخدم المؤشر (Operations Involving Cursors)

يتيح المؤشر ميكانيكية تشغيل السجلات واحدا تلو الآخر عند استخدام أمر الاختيار (SELECT) والذى يختار مجموعة كاملة من السجلات وليس سجلا واحدا فقط. والمثال الآتى يوضح كيفية استرجاع كل معلومات متاحة عن المورد (مثل رقمه وإسمه ورصيده) وذلك لكل الموردين فى المدينة التى يتم تحديدها عن طريق المتغير (Y)

```
EXEC SQL DECLARE x CURSOR FOR
```

```
            SELECT Supplier#, SNAME , ACC
            FROM S
            WHERE CITY = : Y ;
EXEC SQL OPEN x ;
            DO WHILE (more - records - to - come) ;
                EXEC SQL FETCH X INTO :
                Supplierh # , : SNAMEH , :ACCH ;
                ..........
                END ;
EXEC SQL COLSE  x ;
```

ويلاحظ أن أمر الاعلان (DECLARE x CURSOR) يحدد اسم المؤشر (x) ويحدد أيضا المطلوب عن طريق الأمر (SELECT) والذى يتم تنفيذه بمجرد فتح المؤشر (x) بواسطة الأمر (Open).

أما جملة (.. FETCH ... INTO) والتى تظهر فى جملة التكرار(DO ... END) فتستخدم لاسترجاع السجلات واحدا تلو الآخر لاسناد القيم المسترجعة فى المتغيرات المضيفة (Host Values) مثل (... , SUPPLIERH ,SNAMH) ويمكن تكرار جملة الاعلان داخل البرنامج الواحد للتعريف بأكثر من مؤشر.

وفيما يلى ثلاثة جمل تنفيذية تعمل على المؤشر (FETCH , OPEN , CLOSE)

١ - فتح مؤشر (x)

```
EXEC SQL OPEN x ;
```

وفيه يتم وضع المؤشر مباشرة قبل أول سجل فى المجموعة استعدادا لعملية الإسترجاع.

٢ - نقل المؤشر للسجل التالى فى المجموعة ثم إسناد قيم الحقول المسترجعة للمتغيرات المضيفة.

```
EXEC SQL FETCH x INTO host-variable [ , host - variable .... ;
```

وكما سبق الإيضاح فإن جملة (FETCH) تستعمل دائما مع أمر التكرار لعملية إسترجاع السجلات وعند إنتهاء السجلات فى المجموعة فإن تنفيذ جملة (FETCH) يعطى للمتغير (SQLCODE) القيمة (100+) وتتوقف عملية الإسترجاع. ويجدر الإشارة إلى أن عملية نقل المؤشر تتم للأمام وليس للخلف وبواقع سجل واحد ولايمكن أن تتم بواقع ثلاثة سجلات مثلا سواء للأمام أو للخلف.

٣ - قفل المؤشر (x)

```
EXEC SQL COLSE x ;
```

ويمكن بعد عملية قفل المؤشر فتحه ثانية لتشغيل مجموعة سجلات أخرى (وليس بالضرورة نفس مجموعة السجلات) ومن العمليات الأخرى المصاحبة لاستخدام المؤشر عمليات التحديث الحالى (UPDATE CURRENT) والحذف الحالى (DELETE CURRENT) بمعنى أن السجل الحالى الموجود عنده المؤشر يمكن تحديثه أو حذفه حسب الحاجة.

مثـــال

```
EXEC SQL UPDATE S
         SET ACC = ACC + : RAISE
         WHERE CURRENT OF x ;
```

٧ - ٤ لغـــة (SQL) الديناميكيـة

تتضمن لغة (SQL) مجموعة من الأدوات التى تتميز بديناميكية عالية مما يعطى مرونة فائقة وكفاءة ملحوظة عند تشغيل قواعد البيانات بما توفره من إمكانيات التطبيق الفورى أثناء عمليات التداول المختلفة. والتطبيق الفورى (On-Line) يعنى :

- قبول الأمر من الكمبيوتر أثناء التشغيل.
- تحليل الأمر.
- إصدار جملة مناسبة من جمل (SQL) لقاعدة البيانات.
- إظهار رسالة أو نتيجة على الكمبيوتر بعد عملية التنفيذ.

ومن الجمل الديناميكية فى لغة (SQL) جملة الإعداد (PREPARE) والإجراء (EXECUTE) ويمكن إيضاح استخدامهم كالآتى :

```
      DCL SQLSOURCE CHAR(256) VARYIVG ;
EXEC SQL DECLARE SQLOB ] STATMENT ;
      SQLSOURCE = 'DELETE FORM SP WHERE QTY < 100'
EXEC SQLP REPARE SQLOB ] FROM : SQLSOURCE ;
EXEC SQL EXECUTE SQLOB ] ;
```

والمتغير (SQLSOURCE) هو متغير حرفى يقوم البرنامج عن طريقه بتركيب بعض جمل لغة (SQL) (مثل جملة (DELETE) فى المثال السابق) وتسمى جملة المصدر (Source Statement) أما (SQLOBJ) فهو متغير خاص باللغة نفسها ويستخدم لتخزين جمل (SQL) والتى مصدرها (SQLSOURCE) فى شكل كود آلة (Machine Code) ويجدر الإشارة إلى أن الأسماء (SQLSOURCE , SQLOBJ) أسماء اختيارية (Arbitrary). أما جملة الإسناد فى السطر الثالث فهى تسند أمر (DELETE) إلى (SQLSOURCE).

وجملة الإعداد فى السطر الرابع (PREPARE) تأخذ جملة المصدر ثم تترجمها وتربطها للحصول على نسخة مكتوبة للغة الآلة والتى يتم تخزينها فى (SQLOBJ).

وأخيرا فإن أمر الإجراء (EXECUTE) يقوم بتنفيذ النسخة المخزنة فى(SQLOBJ) وبذلك يتم تنفيذ أمر الحذف الفعلى (DELETE).



الفصل الثامن

قواعد البيانات الهرمية

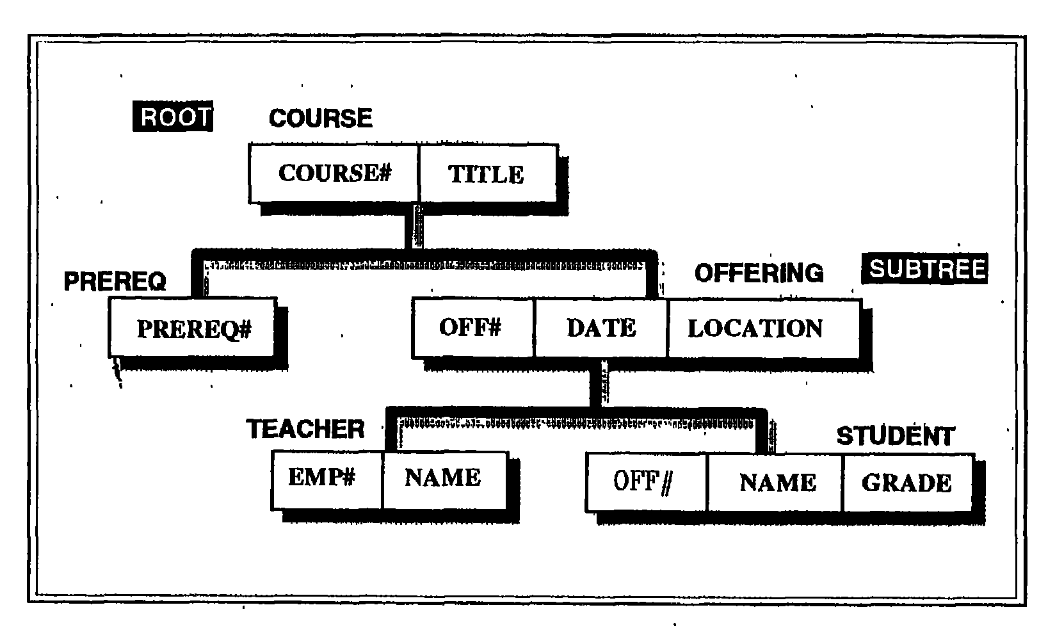
(Hierarchic Databases)



٨ - ١ النموذج الهرمي (The Hierarchic Model)

تتكون قواعد البيانات الهرمية من مجموعة مرتبة من الأشجار (Trees) أو بمعنى أدق ، مجموعة مرتبة من حدوث متعدد (Multiple Occurrences) لشجرة من نوع واحد.

ويتكون النموذج الهرمي من سجل يسمى الجذر (Root) يتفرع منه مجموعة من الشجيرات الفرعية تسمى (Subtrees) ذات مستوى أدنى وتعتمد على السجل الجذر. وهذه المجموعة بدورها تحتوى أيضا على سجل جذر يتفرع منه مجموعة أخرى من الشجيرات الفرعية (Subtrees) ذات مستوى أدنى وتعتمد عليه أيضا وهكذا ، أنظر شكل (٨ - ١) ، أى أن الشجرة كلها تتكون من مجموعة هرمية من السجلات.



شكل (٨ - ١)

والشكل يمثل نموذجا هرميا لقاعدة بيانات تعليمية تخص إحدى الشركات والتى تهتم بإعداد الدورات التدريبية لمستخدميها. وكل حلقة تدريبية (Course) يمكن الالتحاق بها فى عدد محدد من الأماكن داخل كيان المؤسسة التى تتبعها هذه الشركة. وتحتوى قاعدة البيانات على بيانات تفصيلية عن الحلقات التدريبية المتاحة حاليا (OFFERING) والمستقبلية (PREREQ) ويلاحظ أن قاعدة البيانات لها سجل

جذر (Root) هو (Course) واثنين من الأشجار الفرعية والتى لها جذران هما (PREREQ & OFFERING) على التتابع. ويلاحظ أن هذه الأشجار مرتبة حيث تسبق الحلقات الدراسية المستقبلية (PREREQ) الحلقات الحالية (OFFERING) كما يتضح من الشكل.

وبعض الأشجار الفرعية قد تكون جذرا فقط أى لايتفرع منها أشجار فرعية أخرى مثل (PREREQ) وبعضها يتفرع منها أشجار فرعية أخرى مثل (OFFERING) التى تتفرع إلى فرعين من الأشجار عند مستوى أدنى هما (TEACHER & STUDENT) على التتابع. أى أن قاعدة البيانات التعليمية السابقة تتكون من خمسة سجلات هى :

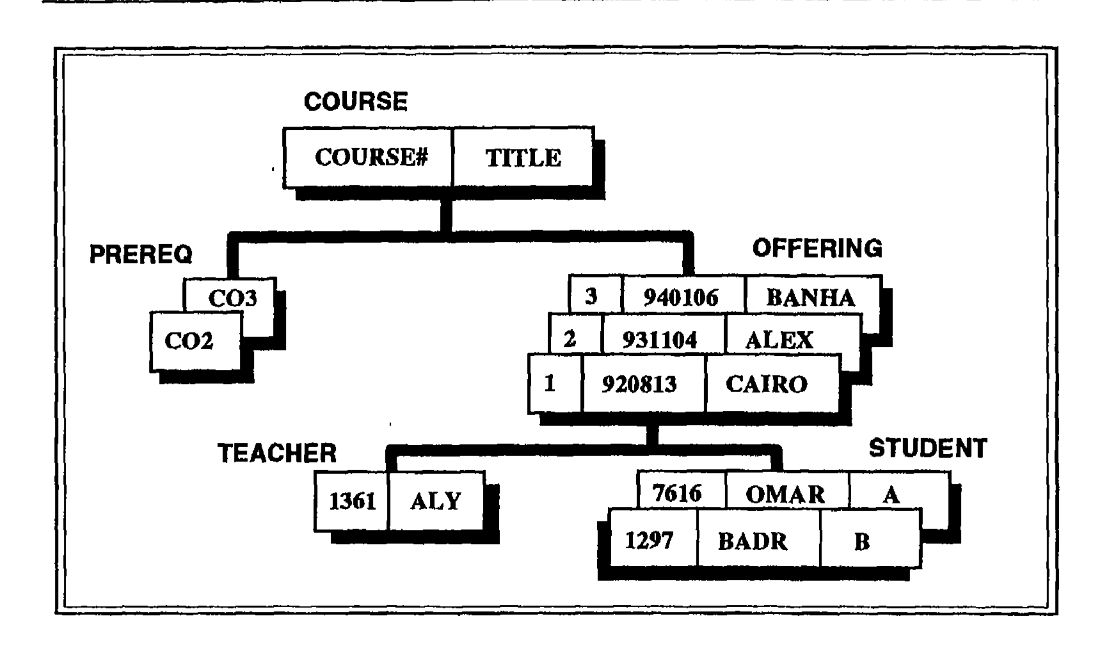
(COURSE, PREREQ, OFFERING, TEACHER & STUDENT)

وأيضا نجد أن السجل (Course) هو أب لكل من السجل (PREREQ) والسجل (OFFERING) والأخير بدوره أب لكل من السجل (TEACHER) والسجل (STUDENT). وبتعريف النموذج الهرمى كما سبق نجد أن الاختلاف الرئيسى بينه وبين النموذج العلاقى (Relatienal) هو أن النموذج الهرمى يمثل المعلومات بنظام رابطة الأب - الإبن أما النموذج العلاقى فيمثل هذه المعلومات عن طريق مفاتيح خارجية (رئيسية وثانوية) وهى عبارة عن حقول مشتركة بين العلاقات. فمثلا العلاقة بين (COURSE , OFFERING) تم تمثيلها برابطة الأب - الإبن أما فى النظام العلاقى فيتم تمثيلها عن طريقة حقل (COURSE) فى السجل (OFFERING). وحدوث النموذج الهرمى (Occurrence) يكون بنفس طريقة تركيبه بمعنى أن حدوث الشجرة يتكون من حدوث سجل الجذر مع مجموعة الأشجار الفرعية المرتبطة به. أنظر شكل (٨ - ٢)

٨ - ٢ تعريف البيانات (Data Definition)

يتم تعريف البيانات عن طريق تركيبين رئيسيين هما توصيف قاعدة البيانات (Database Description) أو (DBD) وبلوك برامج الإتصال (Program Communication Block) أو (PCB).

يقوم الجزء (DBD) بتعريف السجلات والحقول والنظام الهرمى لقاعدة البيانات وهو يماثل (CREATE TABLE) فى النظام العلاقى. والشكل (٨ - ٣) يمثل جزء (DBD) فى قاعدة البيانات التعليمية.



شكل (٨ - ٢)

```
1   DBD     NAME = EDUCPDBD
2   SEGM    NAME = COURSE , BYTES =36
3   FIELD   NAME = ( COURSE# , SEQ) , BYTES = 3, START =1
4   FIELD   NAME = TITLE , BYTES = 33 ,START = 4
5   SEGM    NAME = PREREQ , PARENT = COURSE , BYTES =3
6   FIELD   NAME = PREREQ#, SEQ) , BYTES =3, START =1
    ......................................
    ......................................
```

شكل (٨ - ٣)

وفيما يلى شرح لكيفية قيام (DBD) بتعريف البيانات للقاعدة.

جملة رقم ١ : يتم فيها إسناد إسم (EDUCPDBD) للجزء (DBD) وهو إسم إختيارى.

جملة رقم ٢ : يتم فيها تحديد إسم سجل الجذر (Root) ويشار إليه أيضا بقطاع الجذر (Root Segment) وهو (COURSE) وطوله هو ٣٦ بايت.

جملة رقم ٣ ، ٤ : تعريف بالحقول التى تكون السجل (COURSE) وفيها يتم تحديد الإسم(Name) ، والطول(Length) ومكان البدء من القطاع (Start Position) و (SEQ) تعنى أن حقل (#Course) حقل تتابعى (Sequential) وبالتالى فهو حقل منفرد (Unique) خلال القاعدة.

جملة رقم ٥ : تعرف القطاع (PREREQ) على أنه قطاع مكون من ٣ بايت وأنه إبن (Child) للسجل (Course).

جملة رقم ٦ : تعرف حقل واحد للقطاع (PREREQ) وإسمه (#PREREQ) وهو أيضا حقل تتابعى.

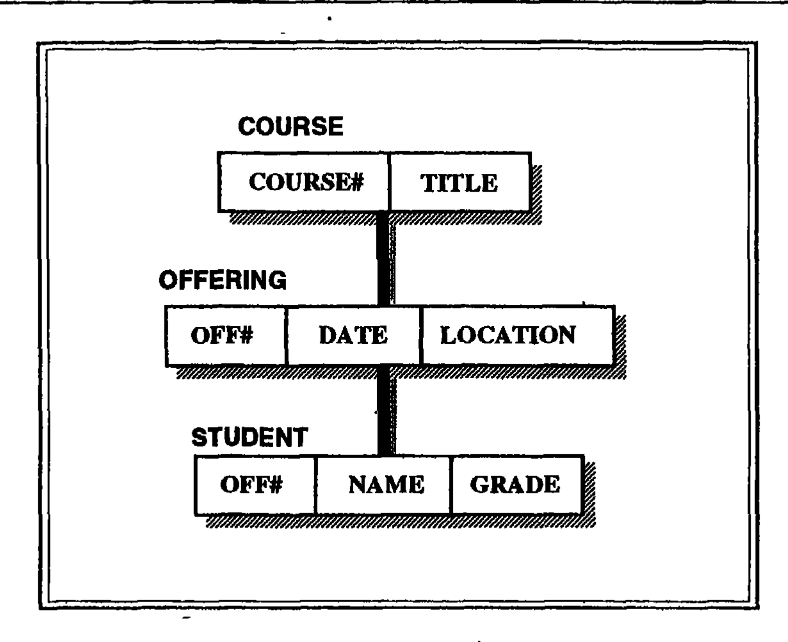
وهكذا يتم تعريف بقية أجزاء قاعدة البيانات.

أما جزء (PCB) فهو يماثل (CREATE VIEW) فى النظام العلاقى أى أنه الجزء الذى يمثل القاعدة حسب طلب المستخدم. أو حسب رؤية المستخدم وبه يمكن رؤية أجزاء من القاعدة وليس بالضرورة القاعدة كلها أى يمكن حذف حقل أو قطاع ومايتصل به من أشجار فرعية أنظر شكل (٨ - ٤)

ويتضح أن النظام الهرمى للجزء (PCB) يجب أن يكون جزءا من النظام الهرمى للجزء (DBD). بمعنى أن السجل الجذر فى (PCB) يجب أن يكون هو نفسه السجل الجذر فى (DBD) وشكل (٨ - ٤) والذى يمثل رؤية المستخدم (User View) يعد من الأشكال الحساسة (Sensitive). حيث أن مستخدم هذه الصورة لايعلم شيئا عن وجود قطاعات أو حقول أخرى لذلك فإن نظام إدارة المعلومات (Information Management System) أو (IMS) يوفر تأمينا وحماية للبيانات.

٨ - ٣ تشغيل البيانات (Data Manipulation)

سوف نستعرض فى هذا الجزء باختصار العمليات الرئيسية التى يمكن بها تشغيل البيانات فى قواعد البيانات الهرمية وتتلخص هذه العمليات فى الآتى :



شكل (٨ - ٤)

١ - الإسترجاع المباشر على سجل فردى معين (GU : Get Unique)
٢ - الإسترجاع المتتابع على سجلات معينة (GN : Get Next)
٣ - الإسترجاع المتتابع على بسجلات لها نفس الأب
(GNP : Get Next within Parent)
٤ - الإسترجاع المتتابع على سجلات لها نفس الأب مع السماح ببعض الإحلال (REPL) أو الحذف (DLET)
Get Hold (GHU,GHN,GHNP), :
٥ - الحشر (ISRT : Insert)
٦ - الحذف (DLET : Delete)
٧ - الإحلال والتحديث (REPL : Replace)

وفيما يلى بعض الأمثلة على استعمال هذه الأوامر مع قاعدة البيانات التعليمية.

١ - الاسترجاع المباشر (Direct Retrieval)

للحصول على أول حلقة دراسية متاحة فى القاهرة يتم كتابة الآتى :

```
GU   COURSE ,
     OFFERING WHERE LOCATION = 'CAIRO' ;
```

٢ - إستدعاء المسار (Path Call)

للحصول على أول حلقة دراسية متاحة (OFFERING) فى القاهرة وإسم الحلقة الأب (Parent Course) يتم كتابة الآتى :

```
GU     COURSE * D ,
       OFFERING WHERE LOCATION ='CAIRO' ;
```

٣ - الإسترجاع المتتابع (Sequential Retieval)

للحصول على كل الحلقات الدراسية المتاحة فى القاهرة يتم كتابة الآتى :

```
GU     COURSE ;
do  until  no  more  OFFERINGS;
       GN  OFFERING  WHERE  LOCATION  =  'CAIRO' ;
end ;
```

٤ - الإسترجاع المتتابع لنفس الأب(Sequential Retrieval within a Parent)

لإيجاد كل الحلقات الدراسية فى القاهرة والخاصة بمادة الكمبيوتر (CO4) يتم كتابة الآتى :

```
GU     COURSE  WHERE  COURSE#  =  'CO4' ;
do  until  no  more  OFFERING  under  Current  COURSE ;
       GNP  OFFERING  WHERE  LOCATION  =  'CAIRO' ;
end ;
```

٥ - الإسترجاع المتتابع عبر قطاعات مختلفة
(Sequential Retrieval Across Segment Types)

لإيجاد أى طالب قام بالتدريس له الاستاذ رقم (1361) يتم كتابة الآتى :

```
GU     COURSE
do  until  no  more  OFFERING ;
       GN  OFFERING ;
       GNP  TEACHER  WHERE  EMP#  =  '1361';
```

```
        if TEACHER found then
        do ;
           GNP STUDENT ;
           leave loop ;
        end ;
end ;
```

ويلاحظ أن إسترجاع البيانات هنا تم عن طريق الإنتقال بين قطاع الإستاذ (TEACHER) وقطاع الطالب (STUDENT) وباستعمال أمر التكرار (Do ... End).

٦ - حشر قطاع (سجل) (**Segment Insertion**)

لإضافة طالب جديد للحلقة الدراسية المتاحة رقم (٨) (OFFERING8) والمسماه (CO4). مع الأخذ فى الإعتبار أن هذا الطالب موظف وله رقم (1409) وليس له درجات (Grade Blank) يتم كتابة الآتى :

```
build new STUDENT Segment in I/O area
        (EMP# = '1409' , GRADE = '  ' ) ;
INSRT COURSE WHERE COURSE # = 'CO4' ;
        OFFERING WHERE OFF# ='8' ,
        STUDENT ;
```

٧ - حذف سجل (قطاع) (**Segment Deletion**)

لحذف القطاع (رقم ٨) الذى تم إضافته فى المثال السابق للحلقة الدراسية (CO4) يتم كتابة الآتى :

```
GHU COURSE WHERE COURSE#='CO4",
        OFFERING WHERE OFF# ='8' ;
DLET;
```

ويلاحظ أن السجل المراد حذفه يجب استرجاعه أولا عن طريق (CHU) حيث يمكن تنفيذ أمر الحذف بعد ذلك وليس قبل الإسترجاع.

٨ - تحديث سجل (قطاع) (Segment Update)

لتعديل المكان المتاح رقم (٤) للحلقة الدراسية (CO4) من القاهرة إلى الإسكندرية يتم كتابة الآتى :

```
GHU COURSE WHERE COURSE# = 'CO4',
        OFFERING WHERE OFF ='4';
change OFFERING Segment in I/O area
        (LOCATION = 'ALEX') ;
REPL ;
```

٨ - ٤ هيكل التخزين (Storage Structure)

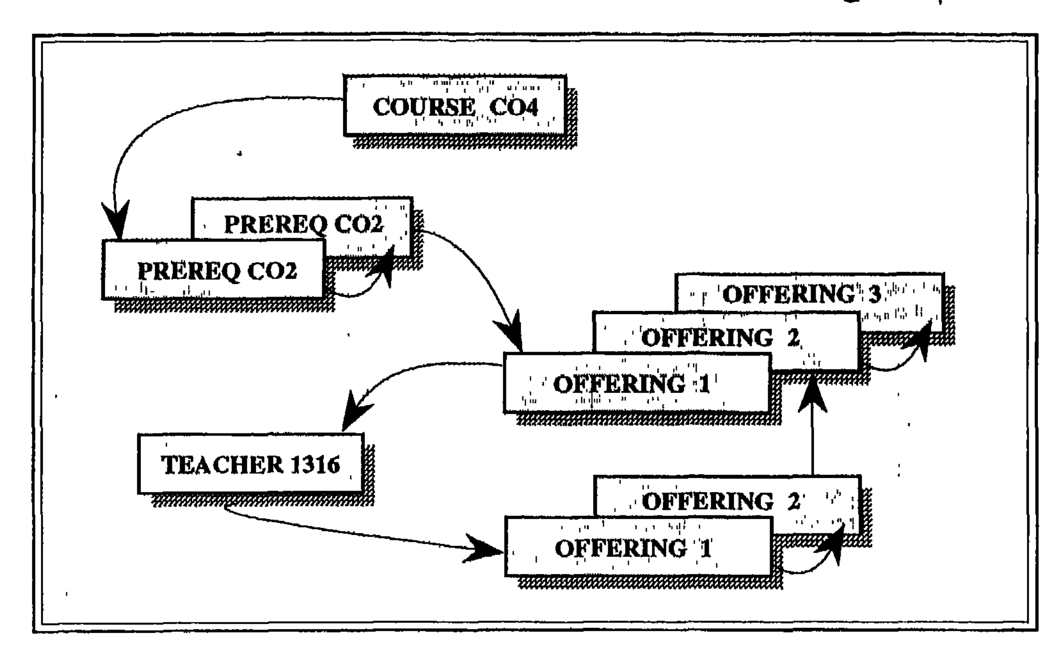
يتم تمثيل قاعدة البيانات الفعلية (Physical Database) بقاعدة بيانات مخزنة (Stored Database) حيث يتم تمثيل كل قطاع (Segment) من القاعدة الفعلية بقطاع مماثل فى القاعدة المخزنة. ويختلف تركيب القاعدة المخزنة من حيث طريقة ربط القطاعات لتكوين النموذج الهرمى المعروف للقاعدة الفعلية ولكن النتيجة النهائية لطريقة الربط تعطى عند تجمعها نفس النموذج الهرمى الفعلى.

وهناك نموذجان رئيسيان من نماذج تراكيب التخزين (Storage Structures) النموذج الهرمى المتتابع (Hierarchic Sequential) أو (HS) والنموذج الهرمى المباشر (Hierarchic Direct) أو (HD) والذى يستخدم المؤشرات (Pointers). ويستخدم النموذج المتتابع عند التشغيل المتتابع وكذلك النموذج المباشر فهو يصلح بكفاءة عند التشغيل المباشر.

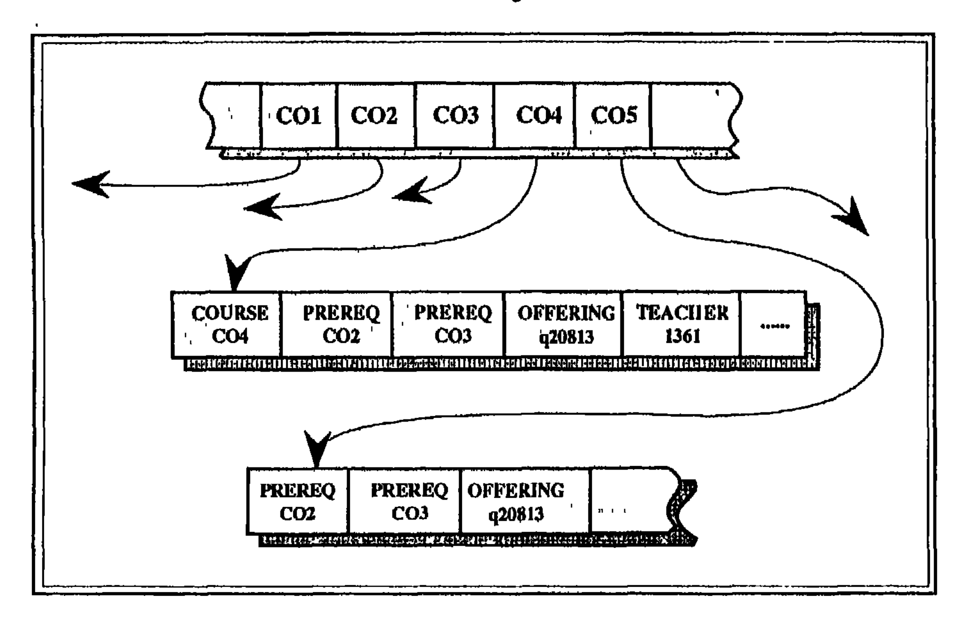
ويساند النموذج المتتابع (HS) طريقتان رئيسيتان للتشغيل هما طريقة التشغيل الهرمية المتتابعة (Hierarchical Sequential Access Method) أو (HSAM) وطريقة التشغيل الهرمية المفهرسة المتتابعة أو (Hierarchical Indexed Sequential Access Method) أو (HISAM).

وكذلك النموذج المباشر (HD) فيسانده طريقتان للتشغيل هما طريقة التشغيل الهرمى لمباشر (Hierarchical Direct Access Method) أو (HDAM) وطريقة التشغيل الهرمى المفهرس المباشر (Hierarchical Indexed Direct Access Method) أو (HIDAM) والشكل (٨ - ٥) يوضح جزءا من قاعدة البيانات التعليمية ممثلة

بطريقة (HISAM). أما شكل (٨ - ٦) فهو يوضح تمثيل جزء من نفس القاعدة باستخدام النموذج الهرمى المباشر عن طريق المؤشرات.

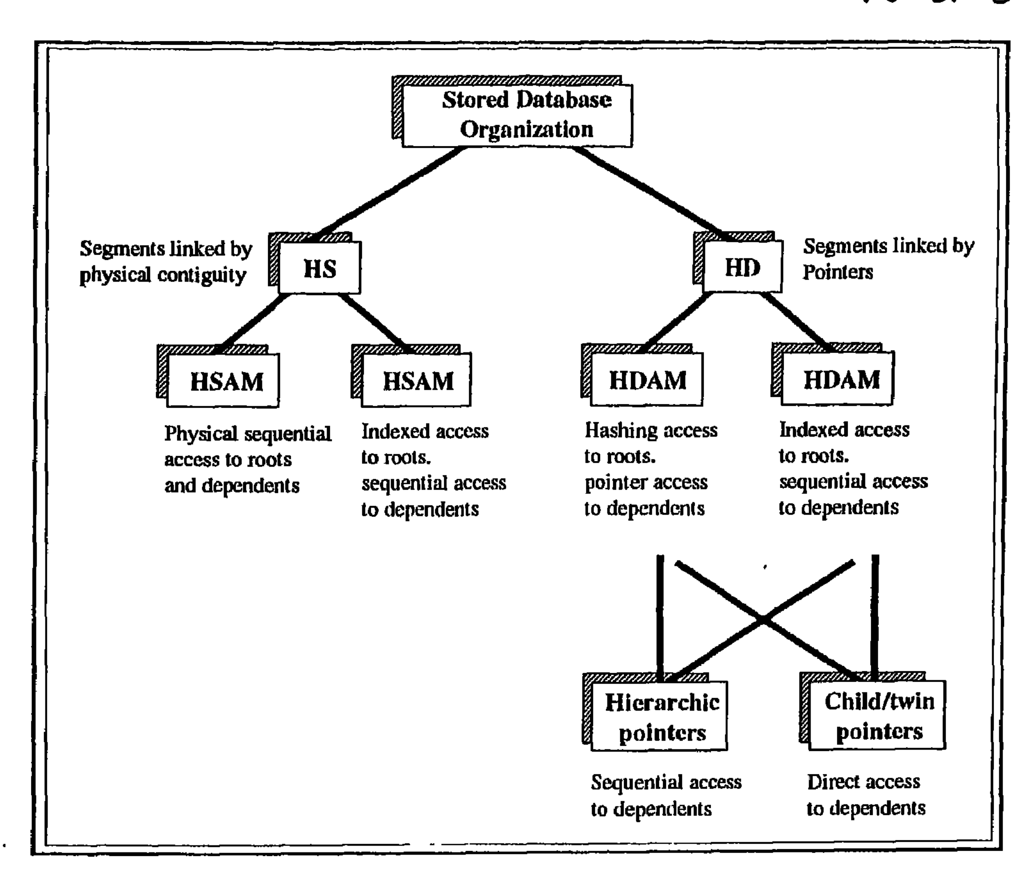


شكل (٨ - ٥)



شكل (٨ - ٦)

أما شكل (٨ - ٧) فهو يوضح هياكل التخزين الأربعة لقاعدة البيانات الهرمية والتى سبق شرحها.



شكل (٨ - ٧)

الفصل التاسع

نظـــام (CODASYL)

كنموذج لقواعد البيانات الهرمية



نظام (CODASYL) من النظم التى تعتمد عليها مجموعة عديدة من قواعد البيانات التجارية والتى غالبا ماتكون من النوع الهرمى. و (CODASYL) هو اختصار (The Conference On Data System Languages) وقد تم تطويره عن طريق (Data-Base Task Group) وتختصر (DBTG). وينقسم نظام (CODASYL) إلى جزئين رئيسيين الجزء الأول هو قسم لغة تعريف البيانات(Data Definition Language) أو (DDL) والتى تصف تركيب قاعدة البيانات والجزء الثانى هو قسم لغة تشغيل البيانات (Data Manipulation Language) أو (DML) والذى يصف كيفية تشغيل قاعدة البيانات.

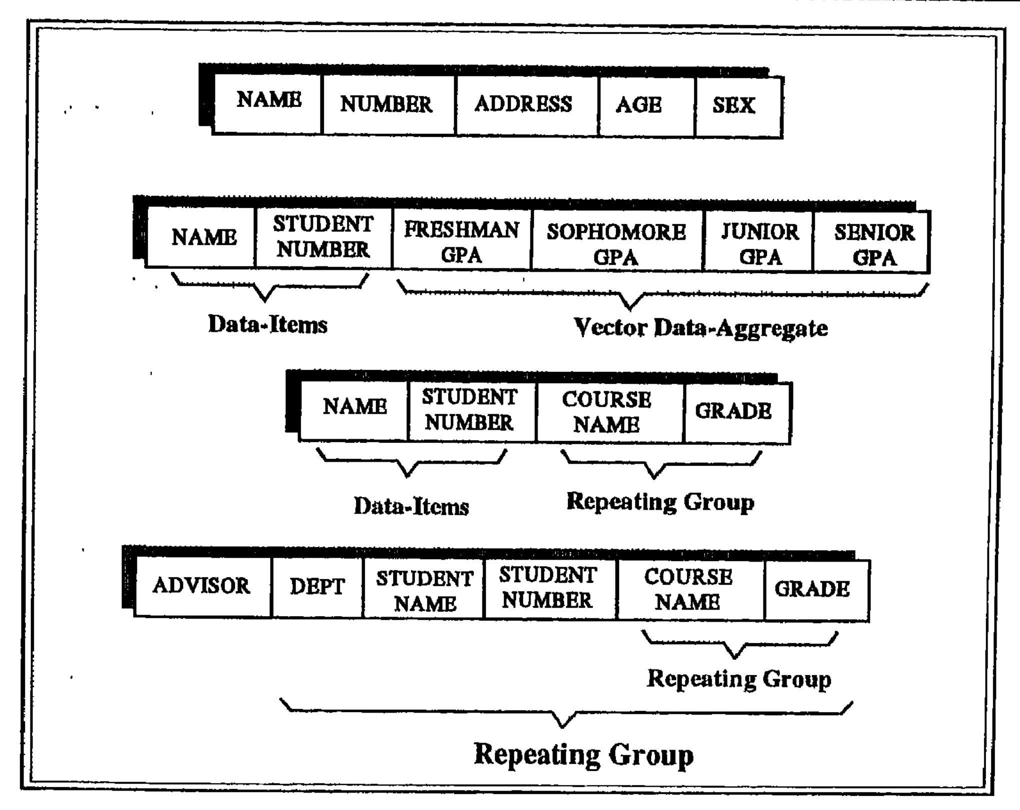
٩ - ١ تعريف البيانات (Data Definition)

تعد وحدة البيانات (Data Item) أصغر عنصر فى (DDL) وهى تقابل ما سبق تعريفه بالحقل (Field) وتحتوى الوحدة على نوع واحد من البيانات ويمكن تجميع وحدات البيانات فيما يسمى تجمع بيانات (Data-Agregate) ويطلق عليها أيضا مصفوفة تجمع بيانات (Vector Data-Agregate) وهى تشبه إلى حد كبير مصفوفة ذات بعد واحد.

ويمكن أن يتكرر حدوث تجمع البيانات أكثر من مرة داخل السجل الواحد فيما يوصف بتجمع البيانات المكرر (A Repeating Group Data-Agregate) ويتكون السجل (Record) من مجموعة من وحدات البيانات ويميز نظام (CODASYL) بين التنظيم المنطقى للسجلات وحدوث هذه السجلات وذلك كما يتضح من الشكل (٩ - ١).

يوضح الشكل (٩ - a١) سجلا مكونا من خمس وحدات بيانات أما الشكل (٩ - b١) فيوضح سجلا مكونا من ٢ وحدة بيانات ومصفوفة تجمع بيانات واحدة والتى تحتوى بدورها على أربع وحدات بيانات والشكل (٩ - C١) يوضح مثالا لمجموعة متكررة من تجمع بيانات ويمكن أن تحتوى المجموعة المتكررة على مجموعة متكررة أخرى بداخلها كما يوضح شكل (٩ - d١).

والفئة (Set) هى تجمع من حدوث السجلات من نوع معين وكل مجموعة لها مالك (Owner) وهو يمثل حدوثا لسجل من نوع آخر وأى حدوث لسجل معين يجعله عضوا (Member) فى هذه الفئة (Set) أو مالكا لها (Owner) ولايمكن لأى سجل أن يكون عضو ومالك فى نفس الوقت.

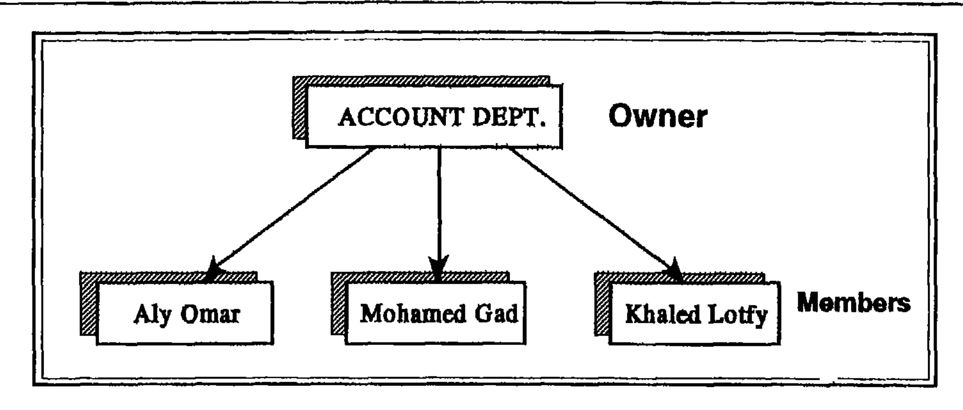


شكل (٩ - ١)

على سبيل المثال نفرض أن قاعدة بيانات لجامعة ما تحتوى على سجل للإدارات المختلفة (Department Record) وسجل للكليات المختلفة (Faculty Record). والفئة التى تجمع بين هذين النوعين من السجلات تربط عضو هيئة التدريس بإدارة معينة (وليكن إدارة الحسابات) وبالتالى تعتبر سجلات إدارة الحسابات هى المالكة (Owner) للفئة وسجلات أعضاء هيئة التدريس هى أعضاء هذه الفئة. أنظر شكل (٩ - ٢).

ويمكن تشغيل السجلات داخل الفئة عن طريق الفهرس (Index) إذا كانت مرتبة أو عن طريق قيمة معينة لحقل ما مثل حقل رقم الطالب لأنه من الحقول غير المكررة والحقل فى هذه الحالة يسمى مفتاح البحث (Search Key).

والنطاق (Realm) هو جزء من قاعدة البيانات والتى تحتوى على حدوث لسجلات معينة.



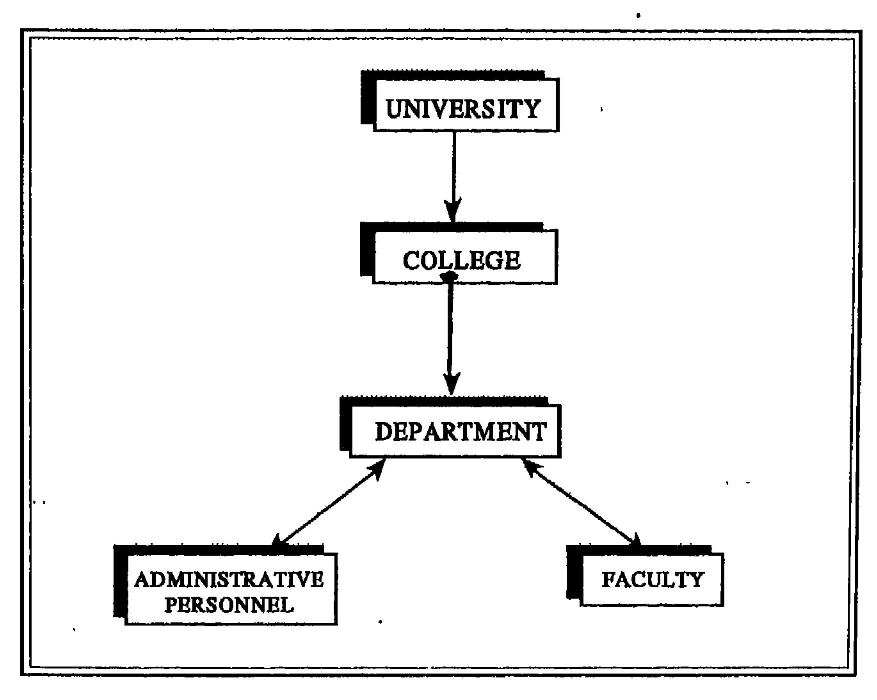
شكل (٩ - ٢)

والنطاق الواحد يمكن أن يحتوى على أنواع مختلفة من السجلات وكل نوع من السجلات يمكن أن يتكرر حدوثه فى أكثر من نطاق. فمثلا قاعدة البيانات التى تحتوى على سجلات الطلبة يمكن أن تحتوى على نطاقين ، نطاق الطلبة الموجودين حاليا ونطاق الطلبة الذين أنهوا دراستهم. وبالتالى يمكن فصل حدوث السجلات الحالية عن السجلات القديمة. كذلك يستعمل النطاق فى فصل البيانات الهامة والتى تستعمل عن طريق أكثر من مستخدم فى أوقات مختلفة. والشكل (٩ - ٣) يلخص العناصر المختلفة المستخدمة لتعريف البيانات فى نظام (CODASYL).

Data Structure	Description
Data-item	Unit of homogeneous data; corresponds to field
Vector data-aggregate repeating group	Array of homogeneous data-itmes. Collection of data-items or agregates occurring multiple times.
Record	Collection of data-items or agregates; logical concept, not a physical one.
Set	Collection of records.
Set member	Record that belongs to a set.
Set owner	Record that identifies a particular group of set members (set occurrence).
Realm	Subset of database records.

شكل (٩ - ٣)

ويقوم نظام (CODASYL) بتمثيل علاقات السجلات داخل قاعدة البيانات الهرمية كما يوضح شكل (٩ - ٤)

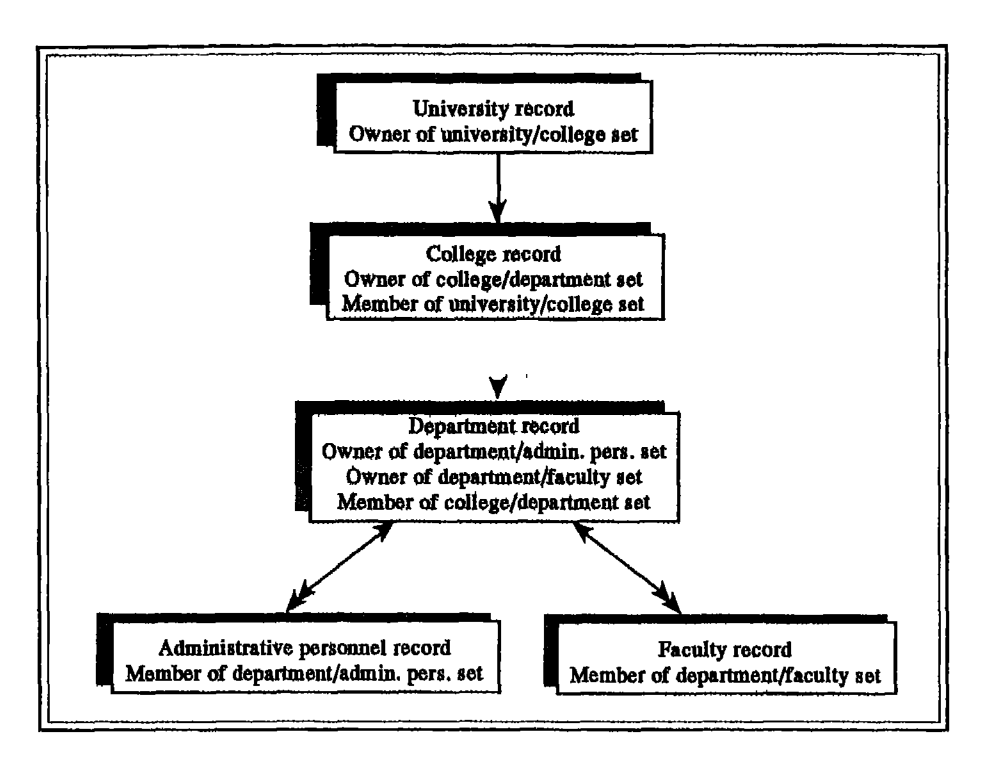


شكل (٩ - ٤)

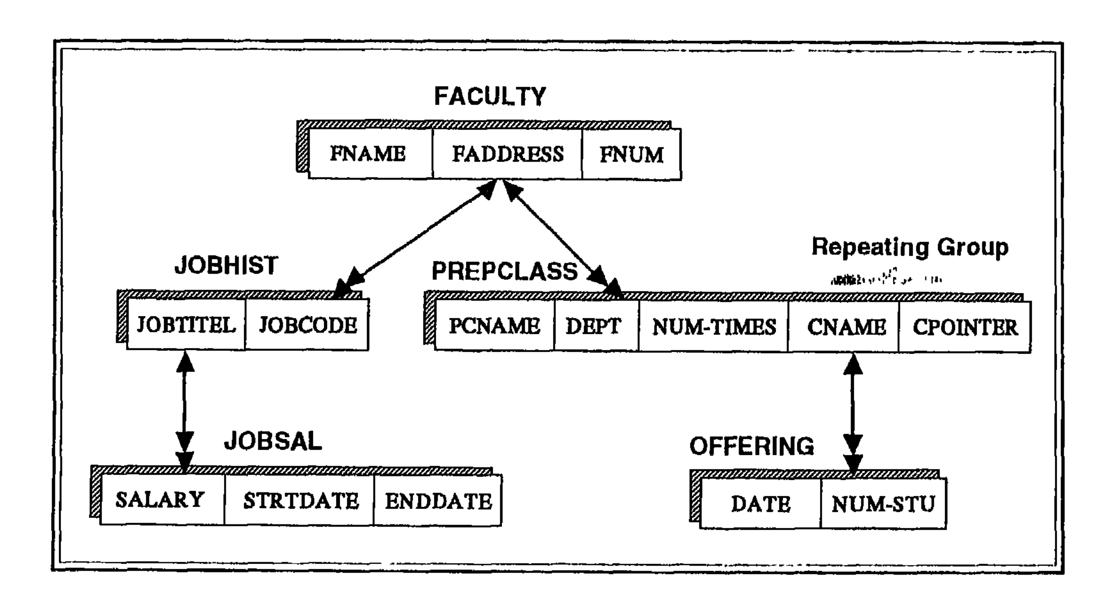
ويلاحظ أن هناك خمسة أنواع مختلفة من السجلات تتصل ببعضها عن طريق أربع علاقات وكل علاقة يمكن تمثيلها بفئة معينة. فمثلا علاقة الجامعة / الكلية تعتبر فئة والكلية / القسم فئة أخرى ... وهكذا.

وبما أن السجل يمكن أن يملك اكثر من فئة لذلك يسهل تمثيل النموذج الهرمى بنظام (CODASYL). فمثلا سجل القسم (Deparment , Personal) فى شكل (٩ - ٥) يملك فئة من سجلات الأشخاص وفئة من سجلات الكلية (Faculty).

وفيما يلى مثال يوضح طريقة تعريف البيانات فى نظام (CODASYL) لقاعدة البيانات الموضحة فى شكل (٩ - ٦)


University record
Owner of university/college set
College record
Owner of college/department set
Member of university/college set
Department record
Owner of department/admin. pers. set
Owner of department/faculty set
Member of college/department set
Administrative personnel record
Member of department/admin. pers. set
Faculty record
Member of department/faculty set

شكل (٩ - ٥)


FACULTY
FNAME
FADDRESS
FNUM
JOBHIST
JOBTITEL
JOBCODE
PREPCLASS
PCNAME
DEPT
NUM-TIMES
Repeating Group
CNAME
CPOINTER
JOBSAL
SALARY
STRTDATE
ENDDATE
OFFERING
DATE
NUM-STU

شكل (٩ - ٦)

وتعريف البيانات لهذا النظام موضح فى شكل (٩ - ٧) ويجب ملاحظة أن أرقام الجمل على اليسار لأغراض الشرح فقط وليست جزءا من عملية تعريف البيانات.

```
1     SCHEMA NAME IS FACULTY-DATA.
2     REALM NAME IS CONFIDENTIAL-SALARY.
3     PRIVACY LOCK IS SESAME.
4     REALM NAME IS OPEN-DATA
5     RECORD NAME IS FACULTY.
6           LOCATION MODE IS CALC USING FNUM
7           DUPLICATES ARE NOT ALLOWED ;
8           WITHIN OPEN-DATA.
9           02 FNAME; TYPE IS CHARACTER 25.
10          02 FADDRESS; TYPE IS CHARACTER 40.
11          02 FNUM; TYPE IS CHARACTER 11.
12    RECORD NAME IS JOBHIST;
13          LOCATION MODE IS VIA FAC-JOBHIST SET;
14          WITHIN OPEN-DATA.
15          02 JOBTITLE; TYPE IS CHARACTER 25.
16          02 JOBCODE; TYPE IS FIXED DECIMAL 2.
17    RECORD NAME IS JOBSAL;
18          LOCATION MODE IS JOBHIST-JOBSAL SET;
19          WITHIN CONFIDENTIAL-SALARY.
20          02 SALARY; TYPE IS FIXED DECIMAL 6.
21          02 STRTDATE; TYPE IS CHARACTER 6.
22          02 ENDDATE; TYPE IS CHARACTER 6.
23    RECORD NAME IS PREPCLASS;
24          LOCATION MODE IS CALC USING PCSAME.
25          DUPLICATES ARE ALLOWED;
26          WITHIN OPEN-DATA.
27          02 PCNAME; TYPE IS CHARACTER 25.
28          02 DEPT; TYPE IS FIXED DECIMAL 3.
29          02 NUM-TIMES; TYPE IS FIXED DECIMAL 2.
```

```
30      02 OFFERING-POINTER; OCCURS NUM-TIMES TIMES.
31        03 CNAME; TYPE IS CHARACTER 10.
32        03 C-POINTER; TYPE IS DATABASE-KEY.
33  RECORD NAME IS OFFERING;
34      LOCATION MODE IS DIRECT O-KEY.
35      WITHIN OPEN-DATA.
36      02 DATE; TYPE IS CHARACTER 6.
37      02 NUM-STU; TYPE IS FIXED DECIMAL 3.
38  SET NAME IS FAC-JOBHIST;
39      ORDER IS SORTED.
40      OWNER IS FACULTY;
41      MEMBER IS JOBHIST PERMANENT AUTOMATIC;
42      ASCENDING KEY IS JOBCODE DUPLICATES ARE
        ALLOWD ;
43      SET SELECTION IS THROUGH LOCATION MODE
        OF OWNER.
44  SET NAME IS JOBHIST-HOBSAL;
45      ORDER IS SORTED ;
46      OWNER IS JOBHIST ;
47      MEMBER IS JOBSAL PERMANENT AUTOMATIC;
48      ASCENDING KEY IS STRTDATE DUPLICATES NOT
        ALLOWED
49      SET SELECTION IS THROUGH LOCATION MODE OF
        OWNER
50  SET NAME IS AC-PREPCLASS;
51      ORDER IS SORTED INDEXED
52      OWNER IS FAC;
53      MEMBER IS PREPCLASS PERMANENT MANUAL;
54      ASCENDING KEY IS PCNAME DUPLICATES NOT
        ALLOWED
55      SET SELECTION IS THROUGH CURRENT OF SET.
56  SET NAME  IS PREPCLASS-OFFERING ;
```

```
57          ORDER IS SORTED;
58          OWNER IS PREPCLASS;
59          MEMBER IS OFFERING TRANSIENT MANUAL;
60          ASCENDING KEY IS DATE DUPLICATES NOT ALLOWED;
61          SET SELECTION IS THROUGH CURRENT OF SET;
```

شكل (٩ - ٧)

وفيما يلى شرح مبسط لشكل (٩ - ٧)

الجملة رقم ١ : تسمية جزء التعريف وهو (FACULTY-DATA).

الجمل رقم ٢-٤ : يتم فيها تعريف نطاقين (Realms) نطاق (CONFIDENTIAL-SALARY) ويستعمل لحماية كلمة السر (SESAME) ونطاق (OPEN-DATA) وهو غير محمى.

الجمل ٥-٩ : تعريف بالسجل (FACULTY). حيث يمكن للبرنامج تحديد مكان السجل عن طريق المفتاح (FNUM) وهو مفتاح منفرد لأن التكرار غير مسموح به وهذا السجل موجود فى نطاق (OPEN-DATA).

الجمل ٩-١١ : تحدد حقول السجل.

الجمل ٣٨-٤٣ : تحدد الشجرة الفرعية المسماه (GET FAC-JOBHIST) وبنفس الطريقة يتم تعريف بقية الأشجار الفرعية.

٩ - ٢ تشغيل البيانات (Data Manipulation)

سوف نستعرض بالأمثلة بعض العمليات الرئيسية التى تستخدم فى تشغيل قاعدة البيانات.

أمر التجهيز والإنهاء (Ready and Finish)

وبـه يتـم تجهيـز النظـام لكـل نطـاق إستعـدادا لعمليـة التشغيـل (READY CONFIDENTIAL - SALARY , OPEN-DATA).

وبعد الإنتهاء من عمليات التشغيل يمكن استعمال أمر (FINISH) لغلق النطاق المفتوح (FINISH OPEN-DATA).

الإسترجـــاع (FIND)

يستعمل أمر (FIND) لاسترجاع سجل معين بحيث يكون هو السجل المطلوب للتشغيل وهو من الأوامر الهامة لأن أمرا مثل (GET) يتعامل فقط مع السجل الواحد أثناء التشغيل ولايستطيع استدعاء سجل آخر.

```
FIND   CURRENT  OF  FACULTY  RECORD
FIND   NEXT  RECORD  OF  FAC-JOBHIST  SET
```

الإحضـــــار (GET)

إسترجاع جزء أو كل السجل الذى تم تجهيزه بالأمر (FIND)

```
GET    FACULTY ; FNUM
```

وهنا يتم الحصول على حقل (FNUM) من السجل (FACULTY).

التعديـــــل (MODIFY)

لتعديل أو تغيير جزء من السجل الحالى

```
MODIFY    JOBHIST
MODIFY    JOBHIST ; JOBTITLE
```

التخزيـــن (STORE)

يستعمل هذا الأمر لإضافة سجل جديد لقاعدة البيانات

STORE FACULTY

الحذف (ERASE)

حذف سجل معين من قاعدة البيانات

ERASE FACULTY

ويجب أن يكون هذا السجل فارغا قبل حذفه وإلا حدث خطأ فى التشغيل.

التوصيل (CONNECT)

توصيل حدوث سجل معين لشجرة فرعية.

CONNECT OFFERING INTO PREPCLASS-OFFERING

الفصل العاشر

قواعد البيانات الشبكية

النظام (IDMS)



يعد نظام قواعد البيانات المتكاملة(Integrated Database Management System) والذى يطلق عليه (IDMS) من النظم التى يمكن تشغيلها على شبكات الحاسب وكذلك على أجهزة الحاسب الكبيرة (Mainframes) باستخدام نظم التشغيل المعروفة مثل (DOS,MVS) وهو أحد الأمثلة المعروفة بنظام (CODASYL) والذى يعنى (Conference on Data Systems Languages) وهى الهيئة المسئولة عن التعريف بلغة الكوبول (COBOL).

ونظام (IDMS) من النظم التى لاتعتمد فى تصميمها على نماذج سابقة التعريف (Predefined) ولكن على العكس يتم تصميم النموذج بعد وقوع الحدث (Event) عن طريق عملية إختصارات (Abstractions) موصفة من قبل المجموعة المكلفة بقاعدة البيانات وتسمى (DBTG) وهى إختصار (Database Task Group) وذلك فى حالة الشبكات (Networks).

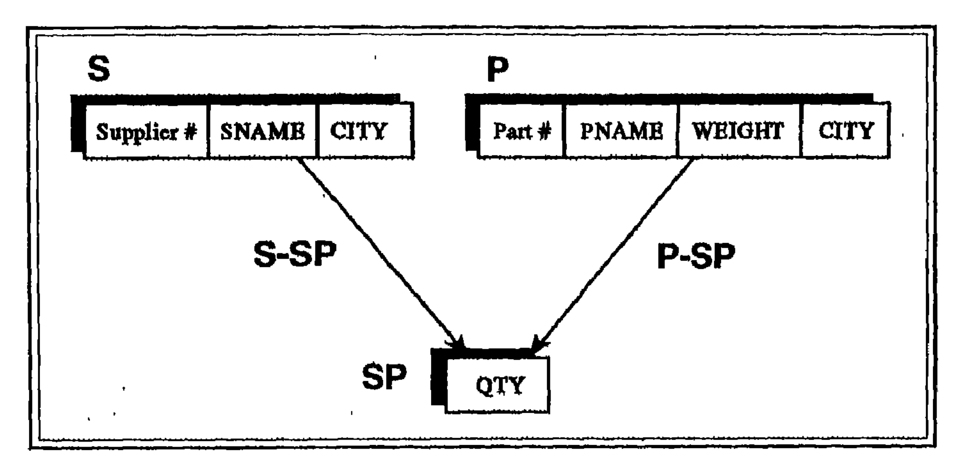
١٠ - ١ النموذج الشبكى (The Network Model)

١٠ - ١ - ١ تراكيب البيانات الشبكية (Network Data Structures)

كما سبق الإيضاح فإن التركيب الشبكى للبيانات يسمح بوجود أكثر من أب (Parent) لنفس الابن (Child). ويتكون النموذج الشبكى من مجموعتين (Sets) ، مجموعة السجلات (Records) ومجموعة الروابط (Links) وكل رابطة لها نوعان من السجلات : سجل الوالد وسجل الابن وكل حدوث لرابطة معينة (Link) يتبعه حدوث منفرد لسجل الأب وحدوث متعدد ومرتب لسجلات الابن. على سبيل المثال فإن علاقة المورد والأجزاء فى قاعدة البيانات يمكن تمثيلها بنموذج شبكى كالموضح فى شكل (١٠ - ١).

ويلاحظ من الشكل وجود نوعين من الروابط (S-SP) ، (P-SP) ونجد أن كل حدوث للعلاقة (S-SP) يتكون من حدوث منفرد للعلاقة (S) مع حدوث منفرد للعلاقة (SP) لكل عملية توريد عن طريق المورد فى العلاقة (S).

وكل حدوث للعلاقة (P-SP) يتكون من حدوث منفرد للعلاقة (P) مع حدوث منفرد للعلاقة (SP) لكل عملية توريد للقطعة والممثل بحدوث العلاقة (P).

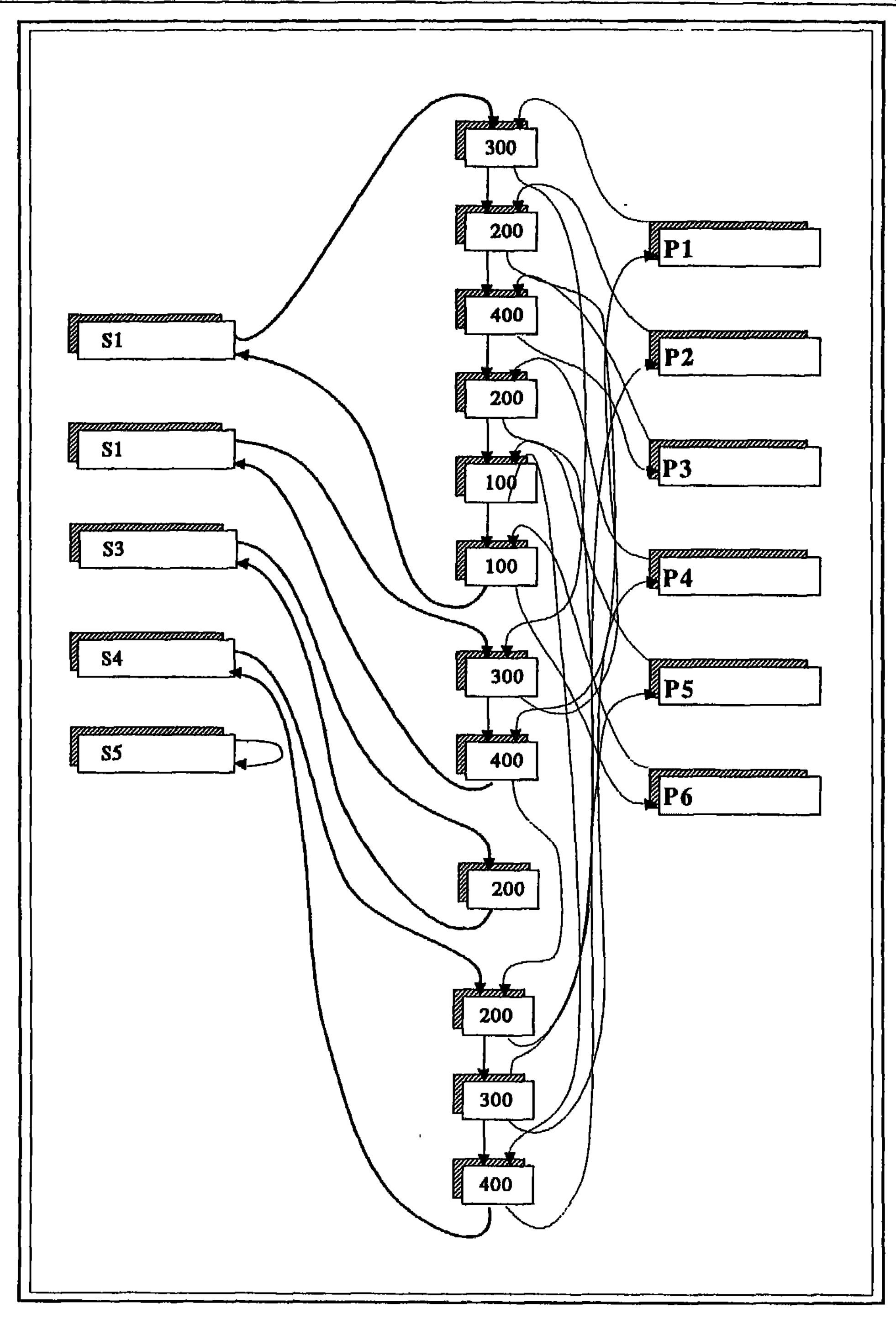


شكل (١٠ - ١)

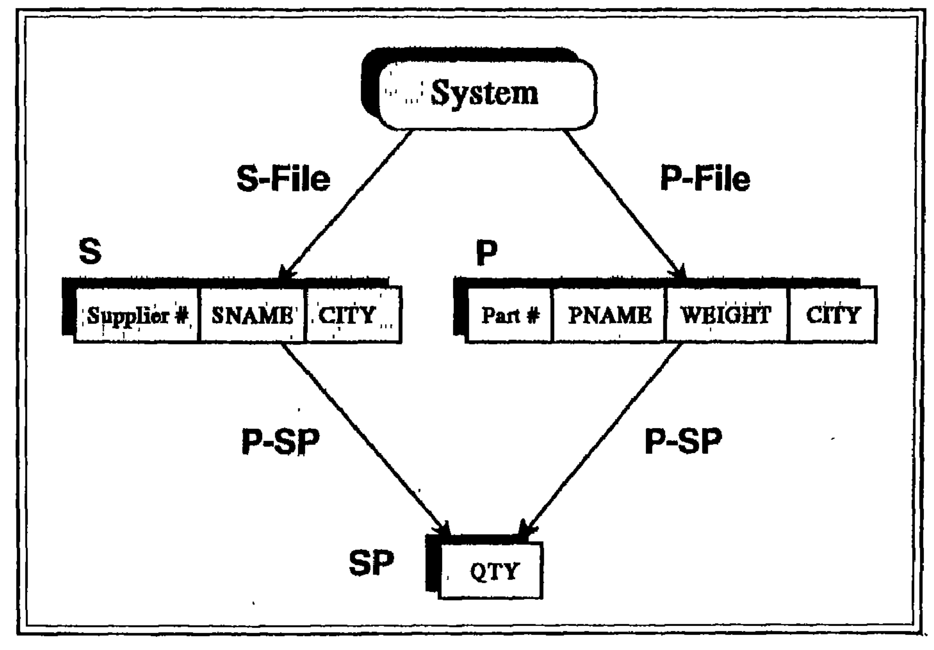
ويوضح شكل (١٠ - ٢) بعض الأمثلة كنموذج لحدوث الرابطتين (S-SP) ، (P-SP).

لاحظ أن لكل حدوث للعلاقة (S) فإن حدوث العلاقة (SP) يظهر بترتيب رقم الجزء، وبالمثل عند حدوث العلاقة (P) فإن حدوث العلاقة (SP) يظهر بترتيب رقم المورد.

ويعتمد تمثيل العلاقات فى شكل (١٠ - ٢) على مجموعة متسلسلة من المؤشرات (Pointers) تنطلق من الأب عند حدوث الرابطة (Link) إلى الأولاد (Childs) عند حدوثهم ثم إلى الأب فى النهاية. ويسمى السجل الذى ليس له أب بالسجل الجذر (Root Record) وفى النموذج الشبكى يعتبر حدوث سجل الجذر إبنا لما يطلق عليه سجل نظام (System Record) ، وتسمى الرابطة عندئذ برابطة النظام (System Link) ورابطة النظام ليست فى الواقع سوى ملف متتابع (Sequential File) ويوضح شكل (١٠ - ٣) الرابطتين(S-File) ، (P-File) كمثال لرابطة النظام.


300
200
400
200
100
100
300
400
200
200
300
400
S1
S1
S3
S4
S5
P1
P2
P3
P4
P5
P6

شكل (١٠ - ٢)



شكل (١٠ - ٣)

١٠ - ١ - ٢ تشغيل البيانات الشبكية

تتكون لغة تشغيل البيانات من مجموعة من المعاملات (Operators) لتشغيل البيانات الممثلة فى صورة سجلات وروابط. ومثال لهذه المعاملات الآتى :

١ - معامل لتحديد سجل معين بمعلومية حقل معين فيه.
مثال تحديد السجل (S) بمعلومية (S1).

٢ - معامل للتحرك بين الأب والابن الأول له فى رابطة ما
مثال التحرك من السجل (S) (للحقل (S1)) للسجل (SP) (للحقل (S1)) عبر الرابطة (S-SP).

٣ - معامل للتحرك من أى ابن إلى الابن الذى يليه فى رابطة معينة.
مثال التحرك من السجل (SP) (الحقل (S1) والقطعة (P1)) إلى السجل (SP) (الحقل (S1) والقطعة (P2)).

٤ - معامل للتحرك من الابن إلى الأب فى رابطة معينة.

٥ - معامل لخلق سجل جديد.
مثال خلق سجل (S4) لمورد جديد.

٦ - معامل لحذف سجل معين موجود فى القاعدة.

٧ - معامل لتحديث سجل معين فى القاعدة.

٨ - معامل لربط سجل معين برابطة معينة.
مثال ربط سجل معين فى العلاقة (SP) بالرابطة (S-SP) أو (P-SP).

٩ - معامل لفصل سجل معين (ابن) من رابطة معينة.
مثال فصل سجل فى العلاقة (SP) من الرابطة (P-SP).

١٠ - ١ - ٣ تكامل البيانات الشبيكة (Network Data Integrity)

مثل بقية النماذج فإن النموذج الشبكى يحتوى على أدوات مبنية فى النظام (Built-in) لتحقيق تكامل البيانات معتمدة على البيانات الأولية فى القاعدة والروابط. فمثلا يمكن تطبيق القاعدة الآتية على كل البنيانات " لايمكن إعتبار أى سجل (إبن) إلا إذا وجد له (أب) ". ويتميز النظام الشبكى بتوفير مستوى جيد من تكامل البيانات مقارنة بالنظم الأخرى.

١٠ - ٢ تعريف البيانات (Data Definition)

من الإصطلاحات شائعة الاستعمال فى تعريف البيانات إصطلاح المخطط (Schema) والذى يعنى ببساطة " تعريف البيانات " أو " وصف البيانات " وتصميم قاعدة بيانات يعنى حقيقة تصميم (Schema). بمعنى أنه لأى قاعدة بيانات يتم تعريف السجلات والحقول والروابط والآباء والأبناء من خلال توصيف المخطط (Schema) لهذه القاعدة. وفى الأمثلة الآتية سوف نستعرض توصيف المخطط (Schema) للقاعدة (المورد - الجزء) فى النظام (IDMS).

١ - إسم توصيف القاعدة (Schema)

SCHEMA NAME IS SUPPLIERS-AXD-PARTS

٢ - تعريف وجود السجل (S)

RECORD NAME IS S

٣ - تحديد مكان السجل (Location) وبواسطته يقوم نظام (IDMS) بتحديد مكان تخزين فى القاعدة لحدوث جديد لسجل معين.

LOCATION MODE IS CALC USING Supplier#

٤ - تحديد حقول السجل (S)

```
02  Supplier#  PIC   X(5)
02  SNAME      PIC   X(20)
02  CITY       PIC   X(16)
```

٥ - تعريف السجل (P)

```
RECORD NAME IS P
LOCATION MODE IS CLAC USING Part#
02   Part#      PIC  X(6)
02   PNAME      PIC  X(20)
02   WEIGTH     PIC  999  USAGE COM-3.
02   CITY       PIC   X(15)
```

٦ - تعريف السجل (S)

ويلاحظ أن تحديد مكان السجل عن طريق الرابطة (S-SP) يعنى أن أى حدوث للعلاقة (SP) يتم تخزينه فعليا من مكان تخزين حدوث العلاقة (S) (مثلا فى

نفس الصفحة أو صفحة قريبة).

```
RECORD NAME IS SP
LOCATION MODE IS VIA S-SP SET
    02  QTY     PIC  999999  USAGE COMP-3.
```

٧ - تعريف العلاقة (S-SP)

```
SET NAME IS S-SP
```

٨ - تحديد ترتيب حدوث السجلات فى العلاقة (SP) عند حدوث الرابطة (S-SP).

```
ORDER IS NEXT
```

٩ - تحديد مالك (Owner) العلاقة (S-SP)

```
OWNER  IS  S
```

١٠- تحديد أنواع السجلات فى العلاقة (S-SP). وهى تحتوى على طريقة الإتصال وهى تكون يدوية (Manual) عن طريق المستخدم وفى بعض الأحوال تكون آلية (Automatic) عن طريق القاعدة.

```
MEMBER IS SP OPTIONAL MANUAL
```

١١- بالمثل يتم تحديد العلاقة (P-SP)

```
SET NAME IS P-SP
ORDER IS NEXT
OWNER IS P.
MEMBER IS SP OPTIONAL MANUAL
```

١٢- تحديد رابطة النظام (System Link) للعلاقة (P) ، (S)

```
SET NAME IS S-FILE
ORDER IS SORTED
OWNER IS SYSTEM
MEMBER IS S MANDATORY AUTOMATIC
                ASCENDING KEY IS SNAME
SET NAME IS P-FILE
ORDER IS SORTED
OWNER IS SYSTEM
MEMBER IS P MANDATORY AUTOMATIC
                ASCENDING KEY IS WEIGHT
```

وبانتهاء تحديد خواص العلاقة المورد - الجزء والمخطط الخاص بها (Schema) يستطيع المستخدم عن طريق الشاشات الخاصة به (User Views) إنشاء تراكيب فرعية (Substructures) من التركيبة الرئيسية السابقة مع مراعاة إمكانية حذف أو إضافة أى حقل أو سجل أو مجموعة بيانات والمثال الآتى يوضح ذلك.

```
ADD SUBSCHEMA NAME IS S-AND-P
     OF SCHEMA NAME IS SUPPLIERS -AND- PARTS
ADD RECORD S
ADD RECORD P
      ELMENTS ARE
      PART#
      CITY.
```

١٠ - ٣ تشغيل البيانات (Data Manipulation)

يشتمل نظام (IDML) على العديد من الجمل التى تستخدم فى عملية تشغيل البيانات لأى قاعدة بيانات شبكية وسوف نستعرض بعض هذه الأوامر مع أمثلة عن كيفية إستخدامها.

١ - جملة البحث (FIND)

لجملة البحث استخدامات عديدة نذكر منها الآتى :

أ - لإيجاد سجل الموردين بمعلومية رقم المورد.

```
MOVE 'S4' TO SUPPLIER# IN S
FIND CALC S
```

ويلاحظ أن أمر البحث (FIND CALC) لايستعمل إلا فى حالة ما إذا كان السجل (S) قد تم تحديد مكانه فى المخطط (Schema) باستخدام (CALC).

ب - لإيجاد المالك (Owner) لحدوث العلاقة (P-SP).

```
FIND OWNER WITHIN P-SP
```

ويلاحظ أن أمر البحث سوف يفشل إذا كانت الرابطة (P-SP) لاتحتوى على رابطة المالك (Owner Linkage).

جـ - لإيجاد رقم جزء لمجموعة أجزاء تم توريدها بواسطة المورد (S3)

```
MOVE 'S3' TO S# IN S
FIND CALC S
FIND FIRST SP WITHIN S-SP
While SP found
PERFOR M
   FIND OWNER WITHIN P-SP
   GET P
   (add Part# IN P to result list)
   FIND NEXT SP WITHIN S-SP
END - PERFORM
```

د - لإيجاد أرقام الأجزاء التى تم توريدها من القاهرة

```
MOVE 'Cario" TO CITY IN P
FIND FIRST P WITHIN P-FILE USING CITY IN P
FPERFORM
    GET P
    (add Part# IN P TO result List)
    (FIND NEXT P WITHIN P-FILE USING CITY IN P
END - PERFORM
```

٢ - جملة التعديل (Modify)

لإضافة (100) إلى حساب المورد (#Acc) رقم (3) فى العلاقة (S)

```
MOVE 'S3' TO Supplier# IN S
FIND CALC S
ADD 100 to ACC3 IN S
MODIFY S
```

٣ - جملة المسح (Erase)

حذف حدوث العلاقة (S) للمورد (S4)

```
MOVE 'S4' TO Supplier# IN S
FIND CALC S
ERASE S [ PERMANENT ]
```

وهناك أوامر عديدة لنظام (IDMS) ولكن لن يتسع المجال لذكرها فى هذا الجزء.

١٠ - ٤ هيكل التخزين (Storage Structure)

فى هذا الجزء سوف نناقش بعض أساليب نظام (IDMS) والمستخدمة فى بناء هيكل التخزين.

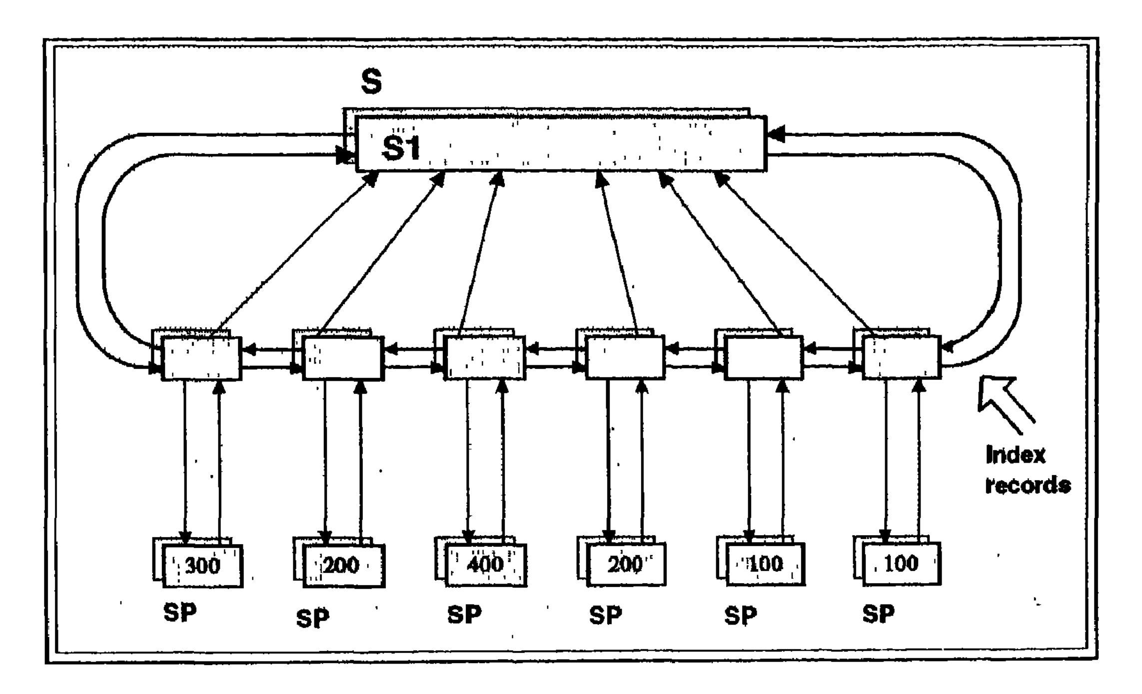
١ - يحتوى كل سجل يتم تخزينه على جزء خفى (Hidden) يضم المؤشرات (Pointers) الخاصة بالمالك (Owner) أو مجموعات البيانات المختلفة ... الخ.

٢ - ينقسم هيكل التخزين إلى نوعين رئيسين حسب تراكيب وحدات البيانات وهما النوع المتسلسل (Chained) والنوع المفهرس (Indexed) ويستعمل لذلك الأمر (MODE IS CHAIN) ، (MODE IS INDEX).

٣ - يستخدم النوع المتسلسل (Chained) فى حالة تخزين وحدات البيانات باستخدام سلسلة من المؤشرات تربط المالك بالعضو الأول (First Member) ثم العضو الأول بالثانى وهكذا. وفى النهاية يتم ربط العضو الأخير بالمالك.

٤ - أما النوع المفهرس (Indexed) فيستخدم فى حالة تخزين وحدات البيانات بنظام الفهرس (Index) وليس بنظام المؤشر المسلسل (Pointer Chain) ويلاحظ من شكل (١٠ - ٤) أن هناك حدوثا كثيرا للفهرس يصاحب حدوث مجموعة (Set) معينة.

٥ - إمكانية تشغيل أى نوع من السجلات عند استخدام حالة التخزين (CALC) فى (نظام (IDMS) ولايكون التشغيل قاصرا على سجلات الجذر (Root Records) كما فى بعض النظم الأخرى مثل نظام إدارة المعلومات (Information Management System (IMS)).



شكل (١٠ - ٤)

الفصل الحادي عشر

نظم إدارة قواعد البيانات

(Database Management Systems)



إن إدارة قواعد البيانات (Database Management) هى المفتاح الرئيسى لنجاح نظم دعم اتخاذ القرارات. فبدون الإدارة الجيدة تفقد بيانات ذات أهمية كبيرة وتضيع معلومات حيوية قد تؤدى إلى قصور فى تدعيم عمل الإدارات أو الهيئات المختلفة. كذلك فإن ضعف إدارة البيانات يؤدى فى الغالب إلى فقد فى مصداقية المعلومات المعتمدة على هذه البيانات ولهذا برزت أهمية قواعد البيانات (Databases) وكيفية إدارة هذه القواعد. ولدراسة أهمية إدارة قواعد البيانات (Database Management) وتأثير ذلك على نظم المعلومات المعتمدة على هذه البيانات فإن هذا الفصل يشرح كيفية إدارة قواعد البيانات بالإضافة إلى البرامج المستخدمة لذلك والتى تسمى نظم إدارة قواعد البيانات (Database Management Systems).

١١ - ١ الخصائص الرئيسية لنظم إدارة قواعد البيانات

من الخصائص الرئيسية لنظم إدارة قواعد البيانات قدرتها على تطوير تراكيب البيانات المختلفة ، وتعريف البيانات وتحديثها وطباعة التقارير وإدارة تراكيب البيانات المخزنة. وفيما يلى بعض هذه الخصائص وشرحها.

أ - بنــاء قاعدة بيانات (Database Creation)

يجب تعريف قاعدة البيانات لنظام إدارة قواعد البيانات حتى يستطيع التحكم فى البيانات وتشغيلها. ويتم تعريف قاعدة البيانات باستخدام لغة تعريف البيانات (Data Definition Langauge) وهى جزء من نظم إدارة قواعد البيانات (DBMS). أيضا عملية إعادة بناء قاعدة بيانات (Re-Creation) من الخصائص التى تميز نظم إدارة قواعد البيانات (DBMS) - فبعض هذه النظم تسمح بإعادة تعريف مجموعة البيانات (Data Sets) التى تم بناؤها ببرامج أخرى حتى يمكن التحكم فيها وتداولها.

ب - تراكيب البيانات

تتيح نظم إدارة قواعد البيانات تراكيب مختلفة للبيانات والتى تسمح بدورها للمستخدم (User) بتخزين البيانات بغض النظر عن الطريقة الفعلية لعملية التخزين. كذلك يقوم النظام بعملية تجميع وحدات البيانات والسجلات المطلوبة للمستخدم. فمثلا إذا طلب أحد المديرين أحدث المعلومات عن رقم المنتج (Product No.) ، وصف

الوحدة (Item Description) ، تكلفة الوحدة (Cost) ، حجم المخزون ، فإنه يطلبها بأى ترتيب ومن سجل واحد أو مجموعة سجلات. وبغض النظر عن كيفية تخزين البيانات (أى ملف يحتوى على هذه البيانات وما إذا كان السجل يحتوى على بيانات أخرى أكثر من المطلوبة أم لا) فإن نظام إدارة قواعد البيانات (DBMS) يتولى تحديد البيانات المطلوبة وتجميعها وتوفيرها للمستخدم.

جـ - تعريف البيانات (Data Definition)

إستقلالية البيانات (Data Independence) تعنى أن البيانات يتم حفظها منفصلة عن برامج تشغيلها. وهذه الخاصية تتيح تغيير البيانات وتحديثها دون التأثير على البرامج المشغلة لها والعكس. فإن المستخدم يستطيع تغيير البرامج دون التأثير على طريقة ترتيب البيانات. وللحفاظ على إستقلالية البيانات فإن نظم إدارة قواعد البيانات تتيح طرقا قياسية (Standard) لتعريف البيانات تتلخص فى أن كل وحدات البيانات يجب أن تحقق طولا قياسيا (Standard Length) وخواص نوعية (Type Specifications).

ويتم تعريف البيانات عن طريق أحد لغات التعريف (Definition Languages) ومنها النموذج الحر (Free Form) والتسلسل الروائى (Narrative) ونموذج كلمة المفتاح(Key Word) ونموذج الفاصل(Separator) وغيرها. والشكل (١١ - ١) يوضح هذه النماذج.

د - الإستفهــام (Interregation)

ويعنى الإستفهام قدرة نظم إدارة قواعد البيانات على إختيار البيانات من القاعدة واسترجاعها لتشغيلها ثم عرض النتائج إما على الشاشة أو فى صورة تقارير.

هـ - التحـديـث (Updating)

تحديث قاعدة البيانات يعنى تغيير بعض أو كل قيم وحدات البيانات أو إضافة بيانات جديدة إليها ولايعنى تغيير تراكيب البيانات (Data Structure). وعملية التحديث تتطلب تحديد البيانات المراد تحديثها وتخزينها فى مكان معين ثم إتباع قواعد التشغيل الخاصة بالتحديث وإعادة القيم إلى قاعدة البيانات.

ومن المشاكل التى تواجه عملية التحديث ما يسمى بنقص التكامل (Lack of Integrity) وهو يعنى أن تحديث أحد الحقول يتم فى بعض الملفات وليس فى جميع الملفات. وهذه المشكلة يتم التغلب عليها عن طريق نظم إدارة قواعد البيانات والتى أحد خصائصها توفير التكامل (Integration) بين جميع الملفات عن طريق ربط الملفات باستخدام حقل فهرسى منفرد (Unique).

DISPLAY THE EFF-DATE FOR PERSONNEL SALARY

free Form

RECHORD NAME IS SALARY IN DATABASE PERSONNEL
DATA ITEM IS EFF-DATE, TYPE IS CHARACTER
LENGTH IS 6 ....

Narrative Form

RECHORD : NAME = SALARY , DATABASE = PERSONNEL
DATA-ITEM : NAME =EFF-DATE, TYPE =CHARACTER
LENGTH = 6 ....

Keyword Form

DATABASE NAME IS PERSONNEL
1. SALARY Y :
2. EFF-DATE(DATE IN 1) , F6 , CHAR ....

Separator Form

شكل (١١ - ١)

بالإضافة إلى ما سبق شرحه من خصائص فإن هناك خصائص أخرى مرتبطة بالعمليات التى تجرى على قاعدة البيانات والتى سيتم شرحها فى الأجزاء التالية.

١١ - ٢ الوظائف الأساسية لنظم إدارة قواعد البيانات

تتمتع نظم إدارة قواعد البيانات بخصائص ومميزات عديدة ، كما سبق الإيضاح ، هذه المميزات تتيح للمستخدم عمليات عديدة وفعالة لتشغيل وحدات البيانات. من هذه العمليات الآتى :

١ - إنشاء قاعدة بيانات جديدة (Creating)
٢ - إضافة بيانات إلى قاعدة البيانات (Appending)
٣ - تصحيح وتعديل البيانات (Editing)
٤ - فرز أو ترتيب البيانات (Sorting)
٥ - البحث عن بيانات محددة (Searching)
٦ - إستخراج التقارير (Reporting)

وسوف يتم توضيح هذه العمليات فى هذا الجزء.

١١ - ٢ - ١ إنشاء قاعدة بيانات جديدة

يقوم نظام إدارة قاعدة البيانات (DBMS) عند إنشاء قاعدة بيانات جديدة بتخصيص مساحة تخزينية على القرص لقاعدة البيانات كما يربط المساحة التخزينية بالبرامج الموجودة فى النظام.

كذلك يتيح النظام للمستخدم توصيف قاعدة البيانات من حيث الملفات والعلاقات (Relations) والحقول داخل كل ملف. وذلك بالإضافة إلى تحديد إسم الحقل وطوله ونوعه (عددى أم حرفى) وتتيح بعض نظم إدارة قواعد البيانات توصيف أنواع أخرى من الحقول مثل حقل التاريخ (Date) وحقل المذكرات (Memos).

١١ - ٢ - ٢ إضافة سجلات جديدة (Appending New Records)

تتيح نظم إدارة قواعد البيانات للمستخدم إضافة سجلات جديدة إلى قاعدة البيانات ويختلف الأمر المستخدم فى ذلك حسب كل نظام. ففى بعض النظم يكون (Insert) وفى بعضها الآخر يكون (Append) ويتم اختيار هذا الأمر من قوائم التشغيل التى تظهر على الشاشة. وتستخدم معظم النظم شاشات إدخال البيانات (Screens) كوسيلة إدخال بيانات السجل.

١١ - ٢ - ٣ تصحيح البيانات (Editing Data)

تتيح نظم إدارة قواعد البيانات إمكانية تصحيح بيانات أى سجل باستخدام الأمر (CHANGE) أو الأمر (EDIT) كذلك يتيح النظام للمستخدم تحديد السجل أو

السجلات المطلوب تعديلها عن طريق المعاملات التى تكتب بعد الأمر وبعض النظم تتيح للمستخدم تنفيذ ذلك عن طريق قوائم الاختيارات التى تظهر على الشاشة. وتتميز نظم إدارة قواعد البيانات بقدرتها على تعديل حقل معين فى عدد من السجلات أو فى كل السجلات فى نفس الوقت.

١١ - ٢ - ٤ فـــرز البيانات (Sorting Data)

معظم نظم إدارة قواعد البيانات تتيح للمستخدم طريقتين لترتيب السجلات. طريقة الفرز (Sorting) وطريقة الفهرسة (Indexing) وطريقة الفرز تؤدى إلى تغيير المواقع الفعلية للسجلات فى الملف وترتيبها حسب بيانات حقل معين يسمى حقل المفتاح (Keyfield). ويتم ذلك عن طريق نسخ الملف بأكمله مع تغيير مواقع السجلات به. وهذا يعنى أن الفرز يتطلب دائما إنشاء ملف جديد بالإضافة إلى الملف الأصلى مما يسبب تحميلا كبيرا على أوساط التخزين المتاحة.

أما الفهرسة فإنها تعتمد على إنشاء فهرس من حقلين فقط أحدهما يحتوى على أرقام السجلات والآخر يحتوى على البيان المطلوب الترتيب بناء عليه مثل الإسم أو تحقيق الشخصية ... الخ. وهذا الحقل يكون مرتبا ترتيبا تصاعديا أو تنازليا. وعند البحث عن إسم معين يتم البحث فى فهرس الإسم عن الإسم المطلوب وبالتالى يتم تحديد رقم السجل المقابل له. وعن طريق رقم السجل يتم استدعاؤه مباشرة.

١١ - ٢ - ٥ البحث عن بيانات محددة (Searching Data)

يلى عملية الفرز (Sorting) عادة عملية البحث (Searching) عن وحدة بيانات محددة. وتتيح نظم إدارة قواعد البيانات البحث باستخدام معادلات منطقية لتحديد شروط البحث. حيث تستخدم المعادلات المنطقية مثل :

AGE > 25 AND WEIGHT < 150
(WEIGHT >70 AND HEIGHT >180) OR SEX = 'F '

والمثال الثانى يعنى البحث فى جميع السجلات عن الأوزان الأقل من (70) والأطوال أكثر من (180) ، وهو مايعنيه المعامل (AND) ، والجزء الأخير من المعادلة يعنى أن حقل الجنس(SEX) يحتوى على الحرف(F) أى أنثى (Femal). والمعامل (OR) يعنى أن أحد الشرطين يجب أن يتحقق. ونتيجة المعادلة الحصول على

بيانات جميع الذكور بالإضافة إلى الأناث الذين تقل أوزانهم عن ٧٠ كجم وتزيد أطوالهم عن ١٨٠ سم.

١١ - ٢ - ٦ طباعة التقارير (Reports)

التقرير (Report) هو قائمة البيانات المطلوب إسترجاعها من قاعدة البيانات على الشاشة أو على الطابعة. ونظم إدارة قواعد البيانات تتيح للمستخدم تحديد مواصفات التقرير المطلوب بدقة مثل عناوين الحقول (Headings) والهوامش (Margins) وعرض الأعمدة (Column Wedith) و ... الخ بالإضافة إلى إمكانية استعمال الشكل القياسى (Standard Format).

وبعض النظم يتيح للمستخدم إرسال التقرير إلى ملف بدلا من الطابعة ويستطيع المستخدم استعمال برنامج معالجة النصوص (Word Processing) لتعديل وتحسين التقرير قبل طباعته.

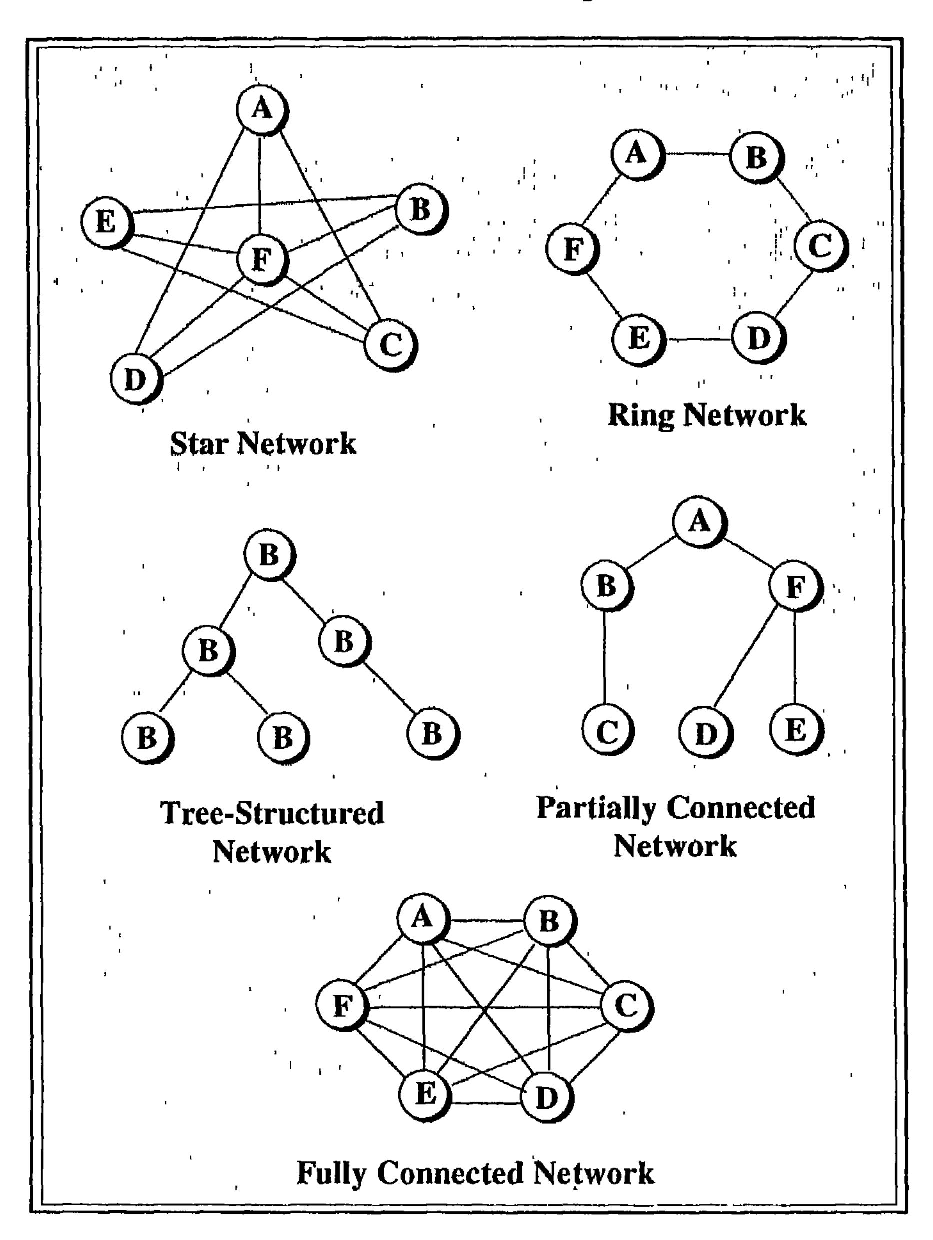
١١ - ٣ قواعد البيانات الموزعة (Distributed Databases)

توفر نظم إدارة قواعد البيانات الحديثة التفاعل الجيد مع المستخدم والكفاءة العالية فى تنفيذ العمليات المختلفة ومن هذه الأنظمة الحديثة نظام قواعد البيانات الموزعة (Distributed Databases). وقاعدة البيانات الموزعة هى قاعدة بيانات واحدة وموزعة على أجهزة حاسب مختلفة موضوعة فى أماكن متباعدة ومرتبطة بشبكة (Network) فمثلا بعض الهيئات والمؤسسات الكبيرة يكون لها فروع منتشرة فى أماكن مختلفة متباعدة لذلك تختار هذه المؤسسات توزيع قواعد البيانات الخاصة بها على حواسب محلية فى الفروع المختلفة بحيث تكون هذه الحواسب مرتبطة بشبكة بدلا من تخزين هذه القواعد فى حاسب مركزى (Central Computer). ومن خلال هذه الشبكة يستطيع المستخدم فى الموقع (A) تشغيل البيانات فى الموقع (B) وذلك بغض النظر عن البعد بين الموقعين والذى قد يكون كبيرا (بين الدول) أو صغيرا (داخل مصنع) ودون الإعتماد على نوع الحاسبات التى قد تكون حاسبات شخصية أو حاسبات كبيرة أو عملاقة.

أيضا من خصائص قواعد البيانات الموزعة مايسمى شفافية المواقع (Location Transparency) والذى يعنى أن المستخدم الذى يطلب بيانات معينة لايحتاج إلى معرفة الموقع الذى تم استرجاع البيانات فيه ولاكيفية توزيع هذه البيانات وتظهر البيانات كما لو كانت قاعدة بيانات واحدة رغم أنها موزعة على المواقع المختلفة.

١١ - ٣ - ١ إختيار الشبكة المستخدمة

هناك أنواع مختلفة من الشبكات (Networks) تختلف بإختلاف طبيعة الربط (Connection) بين المواقع المختلفة كما يتضح من الشكل (١١ - ٢).



شكل (١١ - ٢)

ويمكن شرح هذه الأنواع بإيجاز كما يلى :

١ - الشبكة النجمة (Star Network)

وفيها يتم ربط أجهزة الكمبيوتر فى المواقع المختلفة بجهاز كمبيوتر مركزى وتستخدم فى الشركات التى تتكون من إدارة مركزية وفروع فى جهات مختلفة.

٢ - شبكة الحلقة (Ring Network)

وتكون فيها نقط الربط على هيئة حلقة وهى شائعة الاستخدام فى الشبكات المحلية (Local Area Networks) .

٣ - شبكة التراكيب الشجرية (Tree-Structured Network)

وتستخدم فى النظم التى تحتوى على مستويات هرمية فى الإدارة موزعة فى مواقع مختلفة.

٤ - الشبكة الجزئية الاتصال (Partially Connected Network)

وتستخدم لربط مواقع معينة ببعضها دون البعض الآخر.

٥ - الشبكة الكاملة الإتصال (Fully Connected Network)

يتم فيها ربط كل المواقع ببعضها حيث يتوفر بذلك قدر كبير من المرونة والكفاءة فى إدارة قواعد البيانات.

١١ - ٣ - ٢ طرق توزيع البيانات

بعد اختيار نوع الشبكة المستخدمة فى ربط المواقع يأتى اختيار طريقة توزيع قاعدة البيانات بين الفروع المختلفة ويتم ذلك باستخدام إحدى الطرق الآتية :

١ - تكرار البيانات
٢ - التجزئة الأفقية

٣ - التجزئة الرأسية

٤ - الجمع بين الطرق السابقة.

ولتوضيح كل طريقة نفرض أن شركة طيران معينة لها فروع متعددة. وأحد العلاقات (Relations) الخاصة بهذه الشركة وهى علاقة المسافر (Passenger) وهى موضحة فى شكل (١١ - ٣).

Passenger No.	Passenger Name	Flight No.	Flight Date	Ticket Class
404	Ali G.	337	2/27	Y
607	Ashraf R.	428	1/24	F
212	Loutfy M.	694	3/12	K
312	Mohamed E.	462	2/28	F
617	Tarek M.	428	1/24	Q
915	Ahmed F.	377	2/27	Y
470	Khaled Q.	694	3/12	Y

شكل (١١ - ٣)

١ - تكـرار البيانـات (Data Replication)

فى هذه الطريقة يتم تخزين نسخة منفصلة من العلاقة السابقة فى كل فرع من فروع مكاتب شركة الطيران. وهذه النسخ تتيح عند فقد إحداها الإعتماد على النسخ الأخرى فيما يعرف بالإعتمادية (Reliability). وطريقة تكرار البيانات تتيح سرعة تشغيلها وذلك لأن البحث عن وحدة بيانات معينة يتم داخل الفرع نفسه ولايحتاج إلى الإتصال بالفروع الأخرى.

ومع هذه المرونة والسرعة فى التشغيل فإن هناك مشكلات تظهر عند استخدام هذا النوع. منها متطلبات التخزين العالية لأن كل موقع يحتاج إلى تخزين قاعدة البيانات بالكامل بالإضافة إلى صعوبة تحديث البيانات (Updating) وذلك لأن تحديث أى نسخة يجب أن يصحبه تحديث جميع النسخ الأخرى. لذلك تستخدم

هذه الطريقة عادة عندما تكون البيانات الخاصة بالفروع معظمها للقراءة فقط ولايتم تحديثها كثيرا ... وهذا ينطبق مثلا على دليل التليفون وجداول مواعيد القطارات و ... الخ.

٢ - التجزئــة الأفقيــة (Horizontal Partitioning)

فى هذه الطريقة يتم تقسيم العلاقة إلى عدة علاقات بحيث يحتوى كل منها على الصفوف الخاصة بأحد الفروع فمثلا فى المثال السابق والخاص بشركة الطيران يتم تقسيم العلاقة (Passenger) إلى علاقتين تحتوى كلا منهما على السطور الخاصة بأحد الفرعين كما يوضح شكل (١١ - ٤).

(a) Cairo Branch

Passenger No.	Passenger Name	Flight No.	Flight Date	Ticket Class
404	Ali G.	337	2/27	Y
607	Ashraf R.	428	1/24	F
212	Loutfy M.	694	3/12	K
312	Mohamed E.	462	2/28	F

(b) Port Said Branch

Passenger No.	Passenger Name	Flight No.	Flight Date	Ticket Class
617	Tarek M.	428	1/24	Q
915	Ahmed F.	377	2/27	Y
470	Khaled Q.	694	3/12	Y

شكل (١١ - ٤)

وتستخدم هذه الطريقة عندما تكون حقول البيانات واحدة فى الفروع المختلفة ولكن الحدوث الخاص بالسجلات يختلف من فرع إلى آخر. وتتميز هذه الطريقة بالكفاءة العالية مع توفير المساحة التخزينية فى كل فرع.

٣ - التجزئة الرأسية (Vertical Partitioning)

فى هذه الطريقة يتم تقسيم العلاقة إلى عدة علاقات كل منها يحتوى على الأعمدة الخاصة بأحد فروع الشركة مع ملاحظة وجود حقول مشتركة بين العلاقات تساعد على ربطها ببعضها. أنظر شكل (١١ - ٥)

(a) Flight

Flight No.	Flight Date	Aircraft No.	Pilot ID
337	2/27	18407 T	33461
428	1/24	18495 Q	41309
694	3/12	18746 J	65348
452	2/28	21714 L	62854
428	1/24	21354 L	13149

(b) Passenger

Passenger No.	Passenger Name	Flight No.	Flight Date	Ticket Class
404	Ali G.	337	2/27	Y
607	Ashraf R.	428	1/24	F
212	Loutfy M.	694	3/12	K
312	Mohamed E.	462	2/28	F
617	Tarek M.	428	2/27	Q

شكل (١١ - ٥)

ويلاحظ أن علاقة (Airline) قد تم تقسيمها إلى علاقة الرحلة (Flight) والراكب (Passenger) مع وجود حقول مشتركة بينهما مثل رقم الرحلة (Flight No.) وتاريخ الرحلة (Flight Date) وتستخدم طريقة التجزئة الرأسية بصفة خاصة عندما تكون حقول البيانات المطلوبة فى كل فرع مختلفة عن الفروع الأخرى. فمثلا مكتب حجز رحلات الطيران الخاص بالشركة يعنيه فى المقام الأول تسجيل أسماء الركاب وتاريخ سفرهم وصرف تذكرة الرحلة لهم ولايعنيه رقم الطائرة أو إسم الطيار (Pilot) فى حين يهتم مكتب الشركة داخل المطار ببيانات رقم

الرحلة وتاريخها ورقم الطائرة وإسم الطيار (أو رقمه الشخصى) بغض النظر عن أسماء الركاب أو عددهم أو درجة تذاكرهم (Ticket Class).

٤ - طريقـة الدمـج (Combinotion)

هناك طرق عديدة للدمج بين الطرق السابقة كلها أو بعضها. فمثلا علاقات (Flight) والمسافر (Passenger) فى شكل (١١ - ٥) يمكن دمجها وتخزينها مركزيا (Centrally) أو نسخها فى عدة مواقع. ولكن المبدأ الأساسى الذى يحكم اختيار الطريقة المناسبة وهو تخزين البيانات بالقرب من أماكن استخدامها مع مراعاة تحقيق الأمن (Security) وتكامل البيانات (Integrity) مع تقليل التكلفة.

١١ - ٤ نظم قواعد البيانات الذكية (Intelligent Database Systems)

المقصود بنظم قواعد البيانات الذكية هى النظم التى تستخدم الذكاء الإصطناعى (Artificial Intelligent). وهناك فرعان من فروع الذكاء الإصطناعى يطبقان على قواعد البيانات وهما اللغات الطبيعية (Natural Languages) والنظم الخبيرة (Expert Systems) وسوف نتاولهما بالشرح والتوضيح فى هذا الجزء.

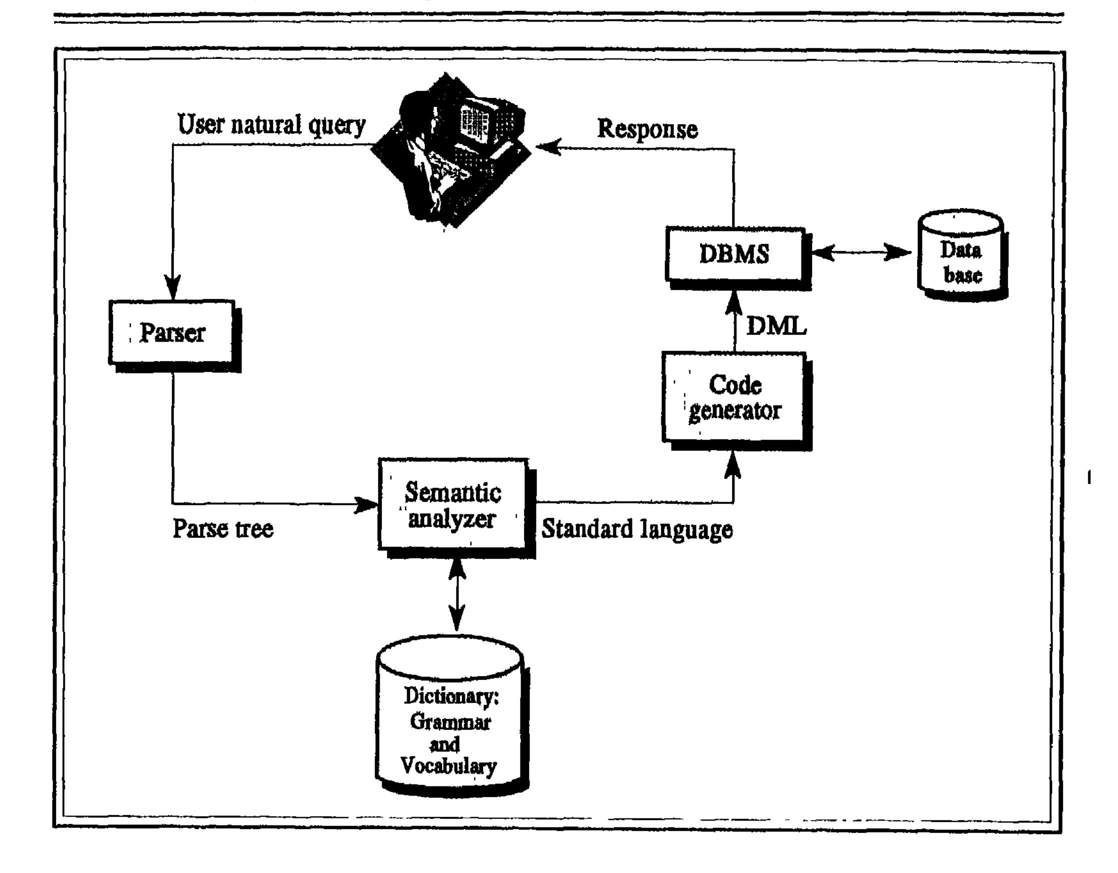
١١ - ٤ - ١ نظم اللغات الطبيعية (Natural Language Systems)

تسمح نظم اللغات الطبيعية للمستخدم بالتحدث مع الحاسب باللغة العادية التى يتعامل بها فى حياته اليومية. فمثلا يستطيع المستخدم كتابة الآتى :

Which pilot will Assign flight No. 337 ?

والمقصود هنا السؤال عن الطيار المكلف بالقيام بالرحلة (337).

وذلك بدلا من استخدام إحدى لغات الحاسب لكتابة الأمر. وتتيح هذه اللغة كفاءة ومرونة عالية للمستخدم العادى غير المدرب على الحاسب على إسترجاع البيانات والتعامل معها وكذلك إمداد المستخدم المدرب بطريقة أسرع وأسهل لتنفيذ العمليات المختلفة على البيانات. والشكل (١١ - ٦) يوضح الأجزاء الرئيسية لنظام اللغات الطبيعية.

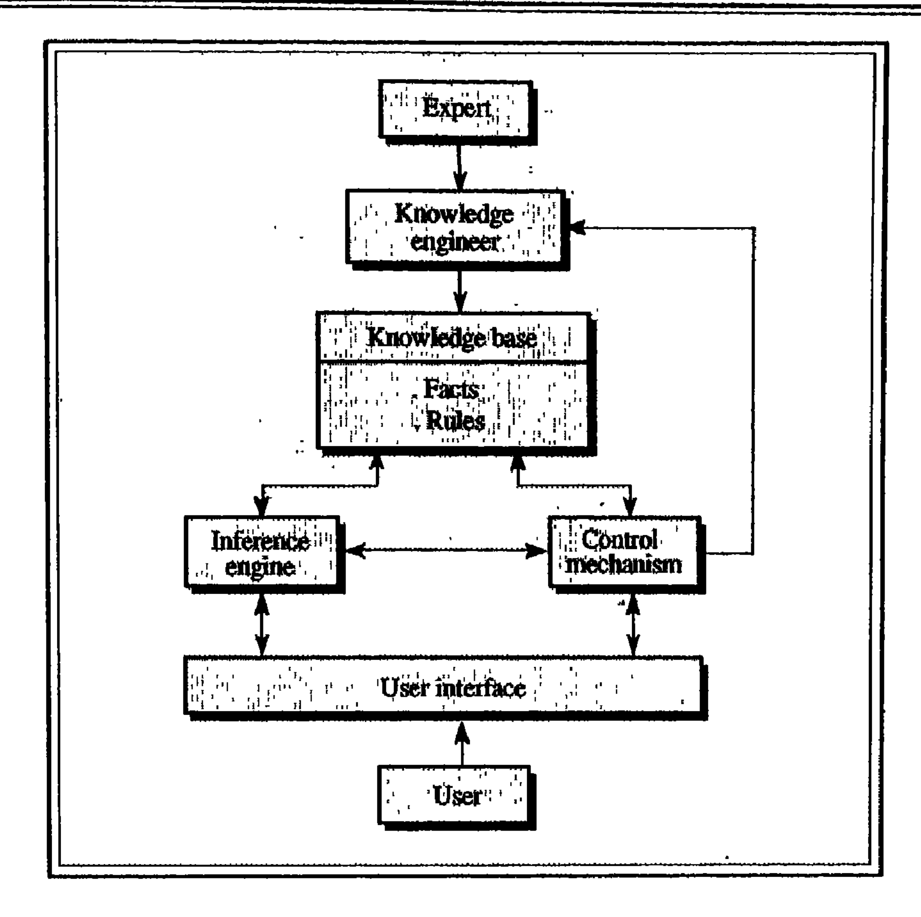


شكل (١١ - ٦)

ورغم ما يصاحب هذه النظم من مشكلات البطء وعدم استجابتها لبعض أسئلة المستخدم إلا أنها استخدمت بكفاءة فى الحاسبات الكبيرة (Mainframes) وأيضا فى الحاسبات الشخصية. وللتغلب على المشكلات المصاحبة لنظم اللغات الطبيعية ظهرت نظم أخرى تتيح التفاعل الجيد بين الحاسب والمستخدم وهى النظم التى تستخدم الرسومات فى التعامل مع المستخدم (Graphic Interface).

١١ - ٤ - ٢ النظم الخبيرة (Expert Systems)

النظم الخبيرة هى التى توظف خبرة ومعلومات الخبراء فى مجال معين لدعم إتخاذ القرار وإيجاد الحلول البديلة المرتبطة بهذا المجال. والأجزاء الرئيسية للنظام الخبير يمكن ايضاحها فى الشكل (١١ - ٧).



شكل (١١ - ٧)

ويلاحظ فى مركز النظام الخبير وجود قاعدة المعلومات (Knowledge Base) وهى تحتوى على الحقائق (Facts) والقواعد (Rules).

والحقائق هى المعلومات التى يتعامل معها النظام الخبير فى مجال خبرته مثل قوانين حركة الأجسام فى مجال الهندسة والقوانين المحاسبية فى مجال المحاسبة ... الخ. أما القواعد (Rules) فهى مجموعة الوسائل والاتجاهات المختلفة المستخدمة فى حل المشاكل المتعلقة بمجال الخبرة المحدد والتى يستخدمها الخبير عادة للوصول إلى القرار السليم. ويتم التعبير عنها غالبا بجمل شرطية (IF-THEN) كالآتى مثلا :

IF INVENTORY < REORDER POINT
THEN PLACE NEW ORDER

وعادة يحتوى النظام الخبير على آلاف القواعد المشابهة.

ومهندس المعلومات (Knowledge Engineer) هو همزة وصل بين قاعدة المعلومات والخبير ووظيفته نقل المعلومات وخبرات الخبير إلى القاعدة بعد تحويلها إلى قواعد (Rules) بالصورة السابق شرحها.

ويحتوى النظام على برنامج يسمى آلة الإستدلال (Inference Engine) وظيفته دعم إتخاذ القرار عن طريق إنتقاء المعلومات المطلوبة. كذلك يقوم نظام التحكم (Control System) بإدارة جميع موارد النظام.

ويتيح النظام الخبير مزايا عديدة لكل من المستخدم والمصمم ومدير قاعدة البيانات ومنها :

أ - يقدم للمستخدم مساعدة كبيرة فى تركيب الجمل التى يستخدمها فى البحث والإستفسار (Query) عن بيانات معينة كما يستطيع إعادة تشكيل الجمل بما يقلل وقت الإستجابة (Response Time).

ب - يساعد مصمم قاعدة البيانات على إنشاء نماذج بيانات جديدة تحقق متطلبات المستخدم.

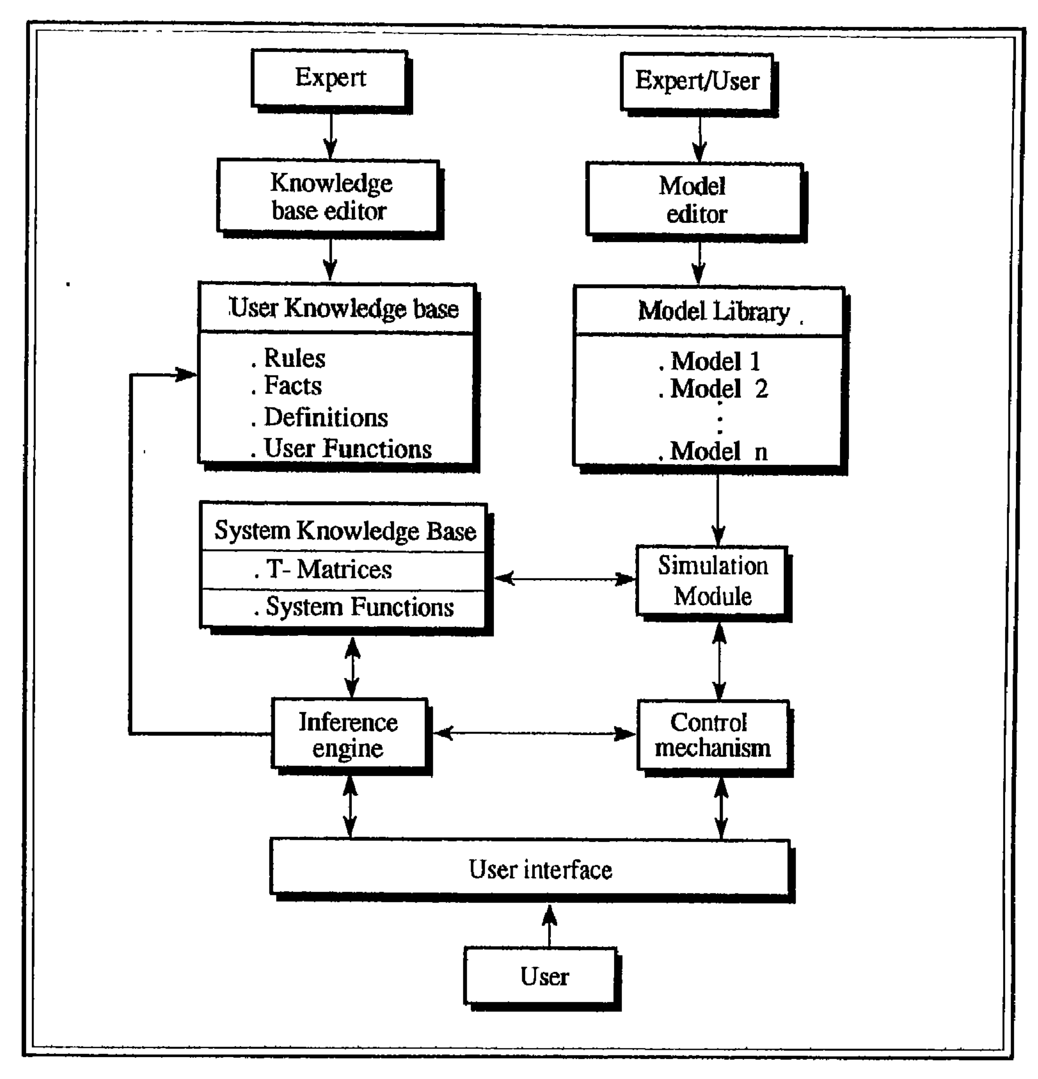
جـ - يحقق أكبر قدر من التأمين (Security) والتكامل (Integrity) للنظام عن طريق تصميم نماذج بيانات تخدم هذا الغرض.

د - القدرة على تقييم أداء قاعدة البيانات وتقديم الوسائل التى تساعد على تحسين هذا الأداء.

١١ - ٤ - ٣ نظم دعم القرار المبنية على المعرفة (Knowledge-Base Decision Support Sytems)

وهى النظم التى تنشأ عن دمج نظم دعم إتخاذ القرار مع النظم الخبيرة وتتكون من مكتبة نماذج (Model Library) وقاعدة معلومات النظام (System Knowledge Base) وقاعدة معلومات المستخدم

(User Knowledge Base) كما يتضح من الشكل (١١ - ٨).



شكل (١١ - ٨)

وتحتوى مكتبة النماذج على مجموعة من نماذج المحاكاة(Simulation Models) يستخدمها المستخدم فى محاكاة نظام الشركة أو الهيئة وتقييم أدائها. ويتم تخزين نتائج المحاكاة فى قاعدة معلومات النظام. أما قاعدة معلومات المستخدم فتحتوى على الحقائق والقواعد والتعريفات والدوال المعرفة بواسطة المستخدم ويتم التفاعل بين هذه القواعد بعضها البعض فى عمليات محكمة تساعد على دعم إتخاذ القرار على أساس من المعرفة والخبرة والتجربة.



الجزء الثانى

نموذج شامل لنظم إدارة قواعد البيانات



الفصل الثانى عشر

مقـــــدمــة



هذا الجزء يوفر معلومات شاملة عن برامج عائلة (DBase) التى تشمل البرامج (FoxPro , FoxBase + , DBase IV , DBase III + , Clipper) باعتبارها أشهر نظم إدارة قواعد البيانات وأكثرها شيوعا. كما أنها تعتبر نموذجا مثاليا لنظم إدارة قواعد البيانات العلاقية (Relational DBMS). وهذا الجزء يتضمن شرحا تفصيليا دقيقا لهذه البرامج وطريقة تشغيلها والقوائم المستخدمة فيها. وقد روعى فى الشرح توضيح خطوات كل عملية يتم إجراؤها بالتسلسل المنطقى الدقيق.

كما يتضمن هذا الجزء أيضا شرح مبادىء البرمجة بصفة عامة ثم تطبيق هذه المبادىء على كتابة البرامج بواسطة برامج عائلة (DBase). بالإضافة إلى شرح شامل لمكونات البرامج متضمنا الأمثلة الواضحة وشرحها التفصيلى.

وهذا الجزء يتكون من ستة وعشرين فصلا يتم من خلالها شرح قوائم المساعد (Assistant) المستخدمة فى برنامج (DBase) وطريقة تشغيل البرنامج من الألف إلى الياء. ثم ينتقل إلى شرح كيفية كتابة البرامج من خلال مشيرة النقطة (Dot Promt) مع شرح كافة الأوامر والدوال المستخدمة.

كما يتضمن الجزء الثالث من الكتاب شرحا تفصيليا لجميع الأوامر والدوال المستخدمة فى برنامج (DBase III+) وكذلك باقى برامج عائلة (DBase).

١٢ - ١ ما هى قاعدة البيانات ؟

رغم أن إسم قاعدة البيانات غير مألوف للإنسان العادى البعيد عن مجال الحاسب ، إلا ان كل إنسان يقابل قواعد البيانات يوميا. فعندما يبحث الإنسان فى دليل التليفون مثلا فإنه يتعامل مع قاعدة بيانات وعندما يسجل بيانات خاصة بالمعارف والأصدقاء فى نوتة معينة للرجوع إليها عند الحاجة فإنه ينشىء قاعدة بيانات. وعندما يبحث موظف معين عن بيانات أحد العملاء عن طريق الدوسيهات الموجودة لديه فإنه يتعامل مع قاعدة بيانات. وهكذا فإن تعامل الإنسان مع قواعد البيانات يأخذ صورا وأشكالا متعددة لايمكن حصرها.

وعندما يقوم موظف الأرشيف بتنظيم البيانات الموجودة لديه فإنه يخصص دوسيها مثلا أو دفترا لكل مجموعة من البيانات التى تخص موضوعا معينا مثل دفتر الحضور والإنصراف ، ودفتر البيانات الشخصية ، ودفتر الشئون المالية ... الخ. كمايقوم الموظف أيضا

بتخصيص صفحة في الدوسيه أو الدفتر لكل موظف بالشركة. وفى صفحة الموظف يقوم بتخصيص أعمدة تمثل بيانات تاريخ الحضور وتاريخ الإنصراف و ... الخ.

وما يحدث مع الحاسب لايختلف كثيرا عن ذلك. حيث يتم إنشاء ملف خاص لكل مجموعة من البيانات التى تخص موضوعا معينا وهو يقابل الدفتر الذى ينشؤه الموظف. ثم يتم إنشاء سجل خاص بكل موظف وهو يقابل الصفحة التى يتم تخصيصها لكل موظف فى الدفتر أو الدوسيه. ثم يتم تسجيل بيانات كل موظف فى السجل الخاص به فى حقل البيانات الذى يمثل كل بيان مطلوب إدخاله.

ولكن هناك اختلافا واضحا بين إنشاء قاعدة البيانات بواسطة الإنسان وإنشائها بواسطة الحاسب. حيث أن الإنسان مثلا يمكنه مباشرة التمييز بين البيانات الموجودة داخل السجل.

فمثلا بالنسبة للإنسان يكون واضحا أن (Mohamed) تمثل إسما وليس رقم تليفون. فى حين لايستطيع الحاسب تمييز ذلك إلا عن طريق وضع قواعد معينة تجعله يستطيع التمييز بين البيانات الحرفية والبيانات العددية مثلا. لذلك فإن إنشاء قاعدة البيانات للحاسب يجب أن يخضع لقواعد معينة. كما يجب أن تتميز هذه القواعد بالوضوح الشديد وذلك لأن الحاسب رغم سرعته الفائقة فى تنفيذ العمليات إلا أنه لايتمتع بأى قدر من الذكاء.

لذلك فإن تصميم هيكل قاعدة البيانات (Structure) يبدأ بتحديد المعلومات المطلوب تخزينها. ثم يتم تقسيم هذه المعلومات إلى وحدات بيانات صغيرة مثل الإسم والعنوان ورقم التليفون و... الخ. وحيث أن هذه الوحدات تكون مشتركة فى جميع السجلات ، لذلك يتم تعريفها للحاسب كحقول. كما يتم تعريف الحاسب بنوع البيانات الموجودة داخل هذه الحقول.

١٢ - ٢ ماهى إدارة قواعد البيانات؟

عندما يبحث الإنسان عن بيان معين داخل قاعدة البيانات أو عندما يقوم بترتيب الأوراق الخاصة بالموظفين داخل الدفاتر حتى يمكنه بسهولة الوصول الى أى موظف فإنه يمارس عملا من أعمال إدارة قواعد البيانات. وبالنسبة للحاسب يحدث نفس الشىء حيث يتم وضع برنامج معين للحاسب ليقوم بإدارة قاعدة البيانات المخزنة به. هذا البرنامج عادة يؤدى المهام التالية :

١ - إضافة بيانات جديدة لقاعدة البيانات.
٢ - ترتيب البيانات بترتيب معين.
٣ - البحث عن بيانات معينة داخل قاعدة البيانات.
٤ - تعديل البيانات.
٥ - عرض وطباعة التقارير.
٦ - مسح البيانات.

وهناك عدة برامج تم تصميمها لإدارة قواعد البيانات أهمها وأكثرها شيوعا وانتشارا هو ما يمكن أن نسميه عائلة (DBase). وهى العائلة التى بدأت ببرنامج (DBase II) ثم (DBase III) ثم (DBase III+) ثم (DBase IV). ثم تلى ذلك ظهور برامج (FoxBase) و(FoxBase +) و(FoxPro) وهى تشتمل على نفس الأوامر والخصائص الفنية الخاصة بعائلة (DBase) كما تضيف إليها أوامر وخصائص جديدة تجعلها أكثر قوة وسرعة. كما ظهرت عائلات أخرى مثل (Oracle) ، (SQL) ، (Informex) ، (4TH Dimension). وهذا الكتاب يتناول شرح خصائص أحد هذه البرامج وهو برنامج (DBase III+) وكذلك النسخة الحديثة منه (DBase IV) وذلك لشعبيته وشيوعه.

١٢ - ٣ برنامج (DBase III+)

هذا البرنامج يمثل النسخة الثالثة من برنامج (DBase) المستخدم مع الحاسبات الصغيرة (Microcomputers) ، حيث سبقته برامج (DBase II) ، (DBase III). وهو يمتاز عن النسخ السابقة بتوفير التفاعل والحوار بين المستخدم والحاسب من خلال القوائم. وهذه القوائم تظهر من خلال برنامج المساعد (Assistant). هذا البرنامج يتيح للمستخدم تنفيذ العمليات المختلفة علي قاعدة البيانات دون الحاجة إلى استعمال الأوامر (Commands) التى تتطلب قدرا كبيرا من الإلمام بالبرنامج.

كما أن برنامج (DBase III+) يمتاز أيضا بإضافة مجموعة كبيرة من الأوامر (Commands) ، والدوال (Functions) ، وأدوات التصحيح (Debugging Tools) التى توفر لمخطط البرامج المزيد من المرونة فى تصميم البرامج الكبيرة.

وبرنامج (DBase III+) ينقسم إلى جزئين رئيسيين وهما برنامج المساعد (Assistant) ومشيرة النقطة (Dot Prompt). وبرنامج المساعد يعتمد على القوائم التى تظهر على الشاشة ويقوم المستخدم باختيار العملية المطلوب تنفيذها. أما مشيرة النقطة

(Dot Prompt) فتتطلب أن يكتب المستخدم الأمر المطلوب تنفيذه والضغط على مفتاح الإدخال.

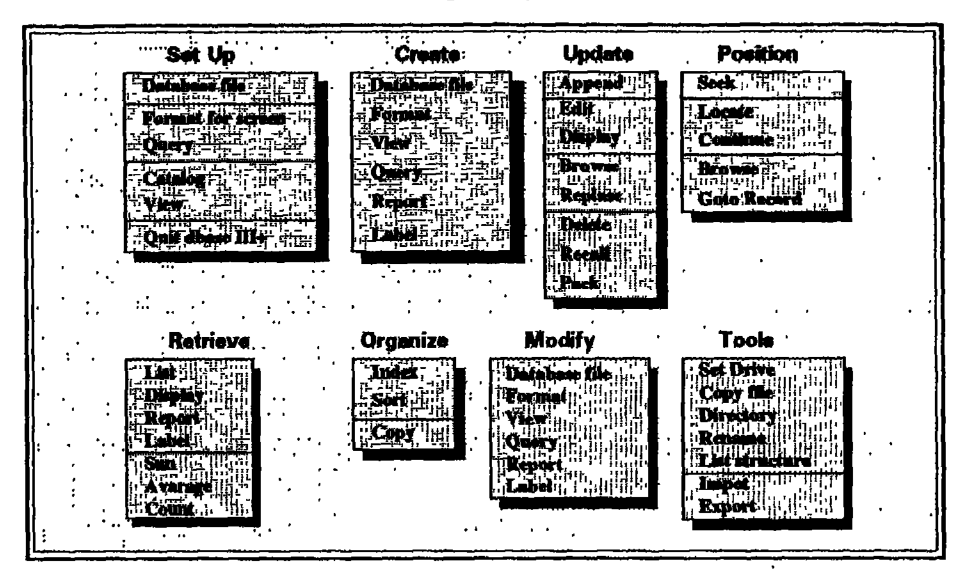
ويلاحظ أنه عند استخدام برنامج المساعد يتم ظهور الأمر الذى يمثل العملية المنفذة عند مشيرة النقطة. وهذا يتيح للمستخدم التعرف على شكل الأمر (Syntax) قبل تنفيذه. وعادة يبدأ أى مستخدم لبرنامج (+DBase III) باستخدام برنامج المساعد. وعندما يكتسب الخبرة الكافية ينتقل إلى استخدام مشيرة النقطة (Dot Prompt) الذى يقوده فى النهاية إلى كتابة البرامج الكبيرة لإدارة قاعدة البيانات.

الفصل الثالث عشر

إنشاء ملف قاعدة البيانات



لتشغيل برنامج (+DBase III) يتم وضع قرص البرنامج فى وحدة الأقراص الأولى ثم كتابة (DBase) وعند بدء تشغيل البرنامج تظهر الشاشة المبينة بالشكل (١٣ - ١). وهذه الشاشة تمثل قوائم برنامج المساعد (Assistant).



شكل (١٣ - ١) قوائم برنامج المساعد (Assistant)

ويلاحظ فى هذا الشكل وجود مجموعة من القوائم (Menus) التى تظهر أسماؤها على الشاشة أعلى كل قائمة. وهذا الشكل يوضح القوائم وهى مفتوحة كلها وهذا للتوضيح فقط. لأن ما يحدث فى الواقع أن القائمة التى يتم وضع المؤشر(Cursor) على إسمها فقط هى التى يتم فتحها أما باقى القوائم فتظل مغلقة. والقائمة التى يتم فتحها يظهر فيها مؤشر يمكن تحريكه حتى يصل إلى الإختيار المطلوب والضغط على مفتاح الإدخال فيتم تنفيذ هذا الإختيار.

١٣ - ١ فتح القائمة

قبل اختيار أى أمر من القوائم يلزم أولا اختيار القائمة التى تحتوى على هذا الأمر. ولتنفيذ ذلك يتم إتباع الخطوات التالية :

١ - يلاحظ فى البداية وقوف المؤشر الخاص بعمود الإختيارات (Menu Bar) على القائمة الأولى (Set Up). وبالتالى تظهر قائمة التجهيز (Set Up) أسفل العنوان

وتحتوى على عدة اختيارات كما يتضح من الشكل السابق.

٢ - يتم الضغط على مفتاح (<---) فيلاحظ تحرك المؤشر الخاص بعمود الإختيارات (Menu Bar) خطوة جهة اليمين. ويلاحظ فتح القائمة الخاصة بالإنشاء (Create).

٣ - يمكن تحريك المؤشر (Cursor) بواسطة مفتاحى السهم يمين (<---) والسهم شمال (--->) لعرض كل القوائم واختيار القائمة المطلوبة.

٤ - عند وصول المؤشر إلى قائمة التجهيز (Set Up) يلاحظ أن الضغط على مفتاح السهم شمال (--->) ضغطة واحدة يؤدى إلى الوصول إلى آخر قائمة يمين الشاشة وهى قائمة الأدوات (Tools).

٥ - يمكن فتح أى قائمة عن طريق كتابة الحرف الأول من إسم القائمة الموجود على عمود الإختيارات (Menu Bar). فمثلا عند الضغط على الحرف (O) يلاحظ ظهور قائمة (Organize).

١٣ - ٢ الإختيار من القائمة

عند فتح أى قائمة يلاحظ أن مؤشر القائمة يكون واقفا على أول اختيار فيها. وبالتالى يظهر هذا الإختيار بالصورة العكسية (Inverse Video) ، أى تكون الحروف فاتحة على خلفية قاتمة عكس الوضع الطبيعى الذى يظهر الحروف قاتمة على خلفية فاتحة. ولتنفيذ أى اختيار من القائمة المفتوحة يتم تحريك العمود الضوئى (Highlight) لأعلى ولأسفل باستخدام السهمين (↑↓) للوصول إلى الإختيار المطلوب ثم الضغط على مفتاح الإدخال.

ملاحظات

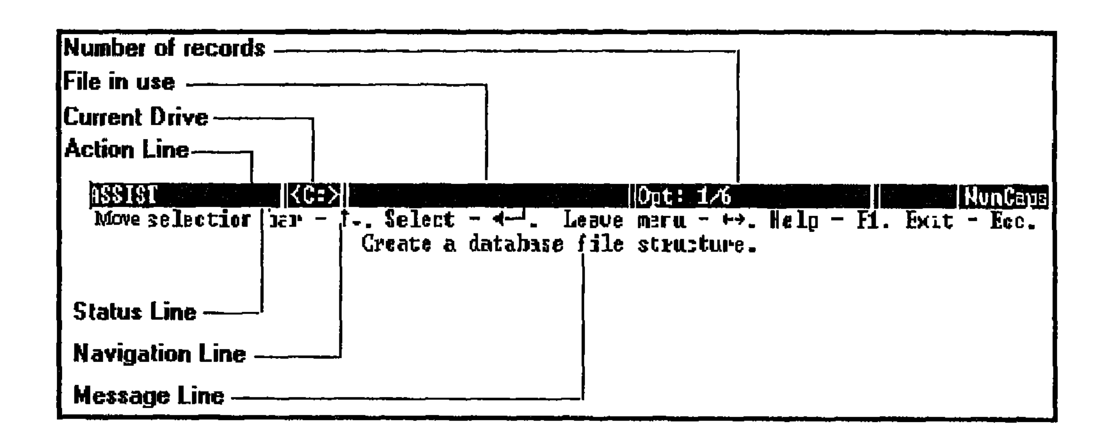
١ - لايمكن الإختيار من القائمة بكتابة الحرف الأول من أى اختيار. ولكن يلزم استخدام مفتاحى الأسهم (↑↓) والضغط على مفتاح الإدخال.

٢ - يلاحظ عند اختيار بعض أوامر أى قائمة ظهور قائمة فرعية أخرى (Submenu) ويتم الإختيار من القائمة الفرعية بنفس الطريقة.

١٣ - ٣ عمود الحالة (Status Bar)

وهو عبارة عن عمود ضوئى (Highlight) موجود أسفل الشاشة كما يتضح من الشكل (١٣ - ٢) وهو يوضح الحالة التى يتم العمل عليها فى البرنامج حيث يبين الآتى :

- إستخدام برنامج المساعد (Assistant) فى التعامل مع البرنامج.
- وحدة الأقراص الجارى العمل عليها.
- ملف قاعدة البيانات المستخدم.
- عدد سجلات الملف ورقم السجل الأول.
- حالة مفاتيح (Num Lock) , (Caps Lock) , (Ins) علما بأن الوضع المبدئى (Default) لهذه المفاتيح يكون (Lowercase) , (Overwrite) (Alphabetic Mode) أى الكتابة بدون إزاحة والحروف الصغيرة والحروف الهجائية على الترتيب.



شكل (١٣ - ٢) عمود الحالة (Status Bar)

تحذير

عند ظهور كلمة (NUM) على يمين عمود الحالة (Status Bar) فإن مفاتيح الأسهم لاتعمل كمفاتيح أسهم ولكن تعمل فى إدخال الأعداد فقط. ويجب الضغط على مفتاح (Num Lock) حتى يمكن تحريك المؤشر باستخدام مفاتيح الأسهم.

ويعرض عمود الحالة معلومات أخرى تعتمد على العمل الجارى تنفيذه. حيث يظهر مثلا معلومات عن السجل الحالى (Current Record) الجارى العمل عليه.

ويلاحظ أيضا أسفل عمود الحالة (Status Bar) وجود سطرين أحدهما يسمى السطر الملاحى (Navigation Line) وهو يوضح المفاتيح المستخدمة لتحريك العمود

الضوئى (Highlight) للقائمة المفتوحة. والسطر الثانى يوضح العمل الذى يجرى تنفيذه فى هذه اللحظة.

كما يلاحظ أيضا وجود سطر آخر فوق عمود الحالة (Status Bar) ويسمى خط الأعمال (Action Line) وهو يوضح الأمر المطلوب تنفيذه. كما يوضح شكل الأمر (Syntax) الذى يمكن بواسطته إدخال الأمر مباشرة عن طريق أوامر النقطة (Dot Commands).

١٣ - ٤ إلغـــاء الأمـــر (Cancelling)

يستخدم مفتاح الهروب (Esc) لإلغاء أى أمر مهما كان عدد الخطوات والقوائم الفرعية التى تم استخدامها. ويتم الضغط على مفتاح الهروب (Esc) عدة مرات للإنتقال من أى قائمة إلى القائمة التى تسبقها.

مــلاحظة

عند الضغط على مفتاح الهروب (Esc) من القائمة الرئيسية يتم الخروج من قوائم البرنامج نهائيا. ويتم الإنتقال من برنامج المساعد (Assistant) إلى مشيرة النقطة (Dot Prompt) وتظهر النقطة (.) فوق عمود الحالة (Status Bar). فإذا أريد العودة إلى برنامج المساعد (Assistant) واستخدام القوائم يتم كتابة الأمر (ASSIST) والضغط على مفتاح الإدخال ، أو يتم الضغط على مفتاح (F2) الذى يؤدى نفس العمل.

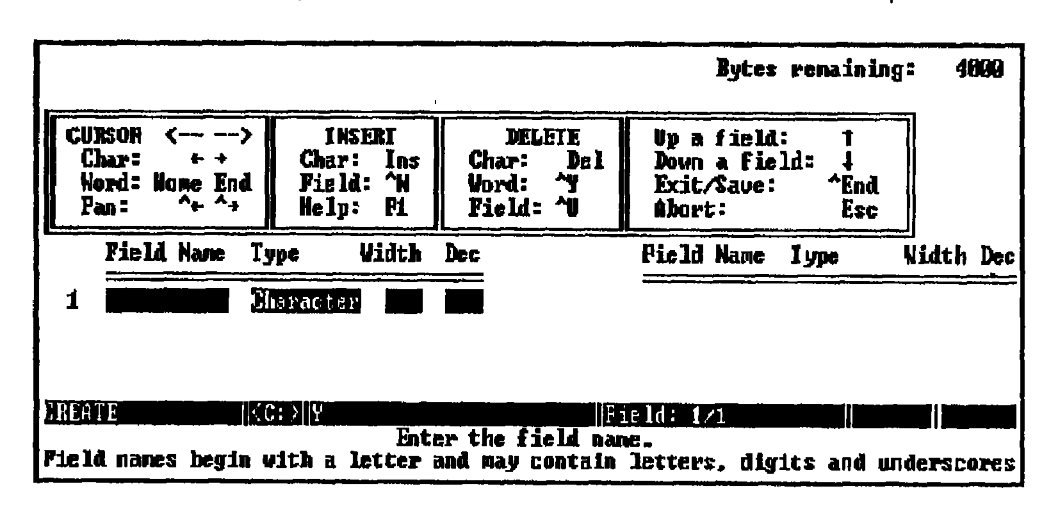
١٣ - ٥ الحصول على المساعدة (Help)

يمكن الحصول على معلومات عن أى أمر من أوامر القوائم عن طريق وضع المؤشر على هذا الأمر والضغط على مفتاح (F1). فى هذه الحالة يظهر مستطيل على الشاشة به معلومات عن هذا الأمر. ثم بالضغط على أى مفتاح يتم الخروج من المساعدة والعودة إلى القوائم مرة أخرى.

١٣ - ٦ إنشاء ملف قاعدة البيانات

لإنشاء ملف قاعدة البيانات يلزم أولا تحديد وحدة الأقراص التى يراد تخزين الملف عليها. لذلك يتبع الآتى :

١ - يتم فتح قائمة الأدوات (Tools).
٢ - يتم اختيار الأمر (Set drive) ، ويلاحظ ظهور الأمر (SET DEFAULT TO) أمام مشيرة النقطة (Dot Prompt) الموجودة أسفل الشاشة فوق عمود الحالة (Status Bar).
٣ - يتم اختيار وحدة الأقراص المراد التخزين عليها والضغط على مفتاح الإدخال.
٤ - يتم فتح قائمة الإنشاء (Create) واختيار (Database File) ثم الضغط على مفتاح الإدخال.
٥ - يتم اختيار وحدة الأقراص المراد تخزين الملف عليها والضغط على مفتاح الإدخال.
٦ - يسأل البرنامج عن إسم الملف فيتم كتابته والضغط على مفتاح الإدخال.
٧ - يلاحظ ظهور الشاشة المبينة فى الشكل (١٣ - ٣) والتى عن طريقها يتم إدخال أسماء الحقول (Field Names) وأنواعها وعرضها (Width) وكل حقل يظهر فى سطر مستقل. ويتم الإنتقال من حقل إلى آخر عن طريق تحريك المؤشر باستخدام مفتاحى (↑↓) ويمكن إدخال حتى ١٢٨ حقلا باستخدام عدة شاشات.
٨ - يتم كتابة إسم الحقل مع مراعاة أن هذا الإسم لايزيد عن عشرة حروف ويجب أن يبدأ بحرف ولايحتوى على أى مسافات بين الحروف. ويمكن استخدام الحروف أو الأرقام أو الشرطة السفلية (Underscore) فى كتابة إسم الحقل.



شكل (١٣ - ٣) شاشة إدخال مواصفات الحقول (Fields)

٩ - يلاحظ وجود مستطيل أعلى الشاشة يوضح للمستخدم المفاتيح التى يستخدمها لتوجيه المؤشر أثناء الكتابة. وهذا المستطيل يمكن إلغاؤه بالضغط على مفتاح

(F1) ، كما يمكن إعادته مرة ثانية بالضغط على نفس المفتاح.

ملاحظات

١ - عند إنشاء هيكل ملف قاعدة البيانات يمكن الكتابة بالحروف الكبيرة (Uppercase) أو الحروف الصغيرة (Lowercase) أو بالجمع بين الحروف الكبيرة والصغيرة. ويلاحظ فى جميع الأحوال أن الكتابة تظهر على الشاشة بحروف كبيرة (Uppercase).

٢ - عند اختيار نوع الحقل يمكن كتابة الحرف الأول من النوع حيث يتم كتابة (C) للحقل الحرفى (Character) ، (N) للحقل العددى (Numeric) ، (D) للحقل التاريخى (Date) ، (L) للحقل المنطقى (Logical) ، (M) لحقل الملاحظات (Memo). كما يمكن الإختيار أيضا عن طريق الضغط على مسطرة المسافات (Space Bar) حيث يتم التحويل من نوع لآخر مع كل ضغطة عليها.

٣ - عند إدخال حقل تاريخى (Date) يلاحظ أن البرنامج يكتب رقم (8) فى خانة عرض الحقل وينتقل المؤشر إلى الحقل التالى. وذلك لأن عرض حقل التاريخ ثابت ويساوى ٨ حروف. كما أن الحقل المنطقى عرضه حرف واحد وحقل الملاحظات عرضه (١٠) حروف.

١٣ - ٧ تخزين هيكل الملف

يجب تخزين هيكل الملف قبل البدء فى إدخال البيانات إليه. ويتم ذلك بالضغط على مفتاح الإدخال بعد ظهور رسالة :

Press Enter to Confirm or Any Other Key to Resume.

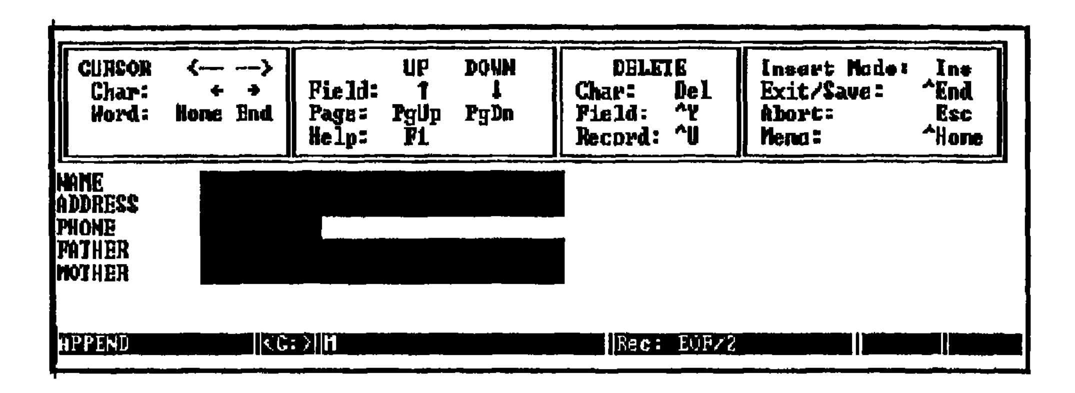
فتظهر رسالة (Wait) لتوضح أن هيكل الملف جارى تخزينه. وبعد إنتهاء عملية التخزين يسأل البرنامج إذا كان المطلوب إدخال بيانات السجلات الآن.

Input Data Records Now ? (Y / N).

فيتم كتابة (Y) لإدخال البيانات.

وفى هذه الحالة تظهر شاشة إدخال خالية كما يتضح من الشكل (١٣ - ٤) بها

أسماء الحقول التى تم إدخالها وأمام كل إسم عمود ضوئى بنفس عرض الحقل الذى سبق تحديده.



شكل (١٣ - ٤) شاشة إدخال خالية

١٣ - ٨ إدخـــال البيـــانـات

يتم إدخال البيانات فى الحقول الظاهرة على الشاشة. وينتقل المؤشر من كل حقل إلى الحقل الذى يليه. ويجب ملاحظة أن الحروف فى هذه الحالة تظهر كما يتم إدخالها. أى أن الحروف الكبيرة (Uppercase) تظهر كبيرة والحروف الصغيرة (Lowercase) تظهر صغيرة. وذلك على عكس ما يحدث عند تكوين هيكل الملف.

وبعد الإنتهاء من إدخال بيانات سجل يظهر السجل التالى. ويتم إدخال بياناته بنفس الطريقة السابقة. وعند الإنتهاء من إدخال بيانات جميع السجلات المطلوب إدخالها يتم الضغط على مفتاحى (Ctrl-End) فى نفس الوقت لتخزين آخر سجل تم إدخاله.

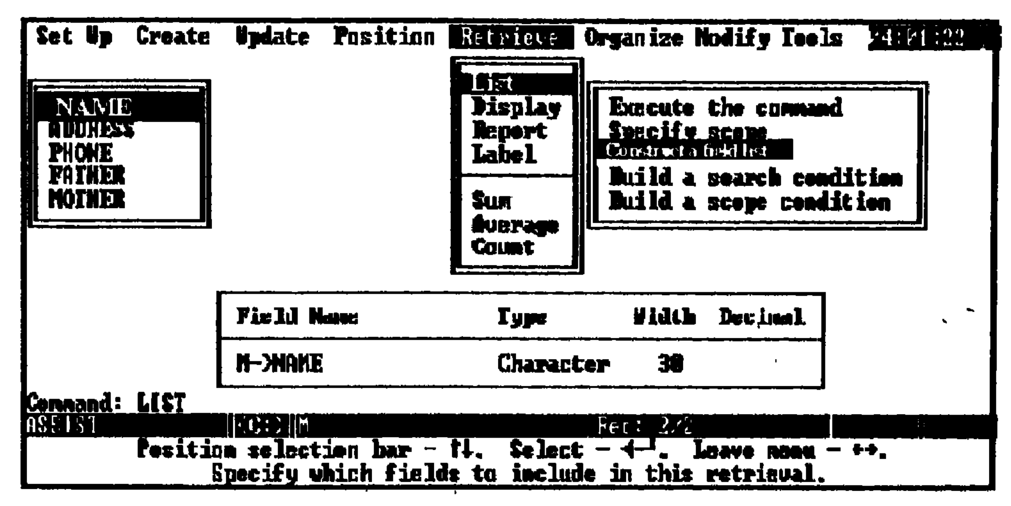
ملاحـظة

يتم تخزين كل سجل آليا عند الإنتقال إلى السجل التالى. أما السجل الأخير فيجب الضغط على مفتاحى (Ctrl-End) لتخزينه.

١٣ - ٩ عرض الملف على الشاشة

عندما يراد عرض ملف قاعدة البيانات والبيانات المخزنة فى كل سجل يتم إتباع الخطوات التالية : انظر الشكل (١٣ - ٥)

الخطوات التالية : انظر الشكل (١٣ - ٥)



شكل (١٣ - ٥)

١ - يتم فتح قائمة الإسترجاع (Retrieve).

٢ - يتم اختيار الأمر (List) من القائمة فيلاحظ ظهور القائمة الفرعية الخاصة بهذا الأمر.

٣ - يتم اختيار الأمر (Construct a Field List) ويلاحظ ظهور قائمة بجميع حقول الملف الذى سبق إنشاؤه فى مستطيل على الجانب الأيسر من الشاشة.

٤ - يتم اختيار الحقول المراد عرض بياناتها وذلك بتحريك المؤشر الموجود فى المستطيل الخاص بالحقول والضغط على مفتاح الإدخال عند كل حقل يراد إدخاله فيلاحظ ظهور علامة (◀) أمام هذا الحقل. وعندما يراد إلغاء حقل سبق اختياره يتم تحريك المؤشر إلى هذا الحقل والضغط على مفتاح الإدخال مرة ثانية فتختفى علامة (◀) أمام هذا الحقل.

٥ - يتم الضغط على مفتاح السهم يمين (<---) للخروج من هذا المستطيل.

٦ - يتم اختيار الأمر (Execute the Command).

٧ - يلاحظ ظهور الرسالة التالية :

Direct output to the Printer? (Y/ N)

فإذا أريد عرض البيانات على الشاشة فقط يتم الضغط على مفتاح الإدخال لأن الوضع المبدئى (Default) هو (N) أى أنه لايراد طباعتها ولكن عرضها على الشاشة فقط.

٨ - يلاحظ ظهور بيانات الملف على الشاشة كما يتضح من الشكل (١٣ - ٦).

٩ - أما إذا أريد طباعة البيانات على الطابعة فيتم كتابة (Y) ويجب فى هذه الحالة التأكد من أن الطابعة قد تم تشغيلها (ON) وأنها موصلة بالجهاز.

```
 Set Up  Create  Update  Position  Retrieve  Organize Modify Tools  14:10




Record#  NAME                                  ADDRESS              PHONE
 FATHER                                 MOTHER
      1  mohamed hasan fathy                   12-ain shams         56526756
 hasan fathy                            samya tawfik
      2  ahmed soliman tarek                   40-ahram-street      6789889
 soliman tarek                          faten kamal
ASSIST            ||<C:>||M                   ||Rec: 2/2            ||
                   Press any key to continue work in ASSIST.
```

شكل (١٣ - ٦) عرض بيانات الملف على الشاشة


Converted by Tiff Combine - (no stamps are applied by registered version)

الفصل الرابع عشر

إنشاء شاشات الإدخال



تم فيما سبق شرح كيفية إدخال البيانات إلى ملف قاعدة البيانات باستخدام الشاشة المستخدمة فى البرنامج والتى تماثل هيكل الملف الذى تم تكوينه. وعادة تكون هذه الشاشة غير واضحة للمستخدم كما أن أسماء الحقول تكون غير مفهومة وتحتاج إلى كثير من الشرح والتوضيح للمستخدم.

لذلك فإن برنامج (+ Dbase III) يتيح تصميم شاشة إدخال واضحة ومفهومة وذلك عن طريق مايسمى راسم الشاشة (Screen Painter). ويمكن استخدام هذه الشاشة فى إدخال البيانات وكذلك فى عرض البيانات على شاشة الحاسب.

ويتم تكوين هذه الشاشة عن طريق إتباع الخطوات التالية :

١ - يتم فتح قائمة الإنشاء (Create) واختيار (Format) منها.

٢ - يتم اختيار وحدة الأقراص التى يتم فيها تخزين الملف ويلاحظ ظهور الرسالة التالية :

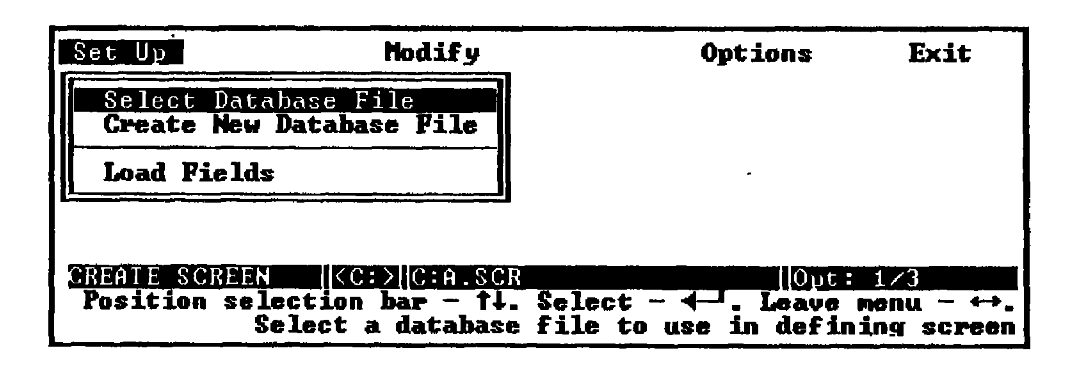
Enter the Name of File

٣ - يتم كتابة إسم ملف شاشة الادخال المطلوبه وليست هناك حاجة الى كتابة الامتداد (Extension) لأن برنامج (+ Dbase III) يضيف الإمتداد (scr.) آليا.

٤ - يلاحظ ظهور قوائم تصميم شاشة الإدخال وسوف يتم شرحها فى الأجزاء التالية.

١٤ - ١ قائمة تصميم شاشة الإدخال

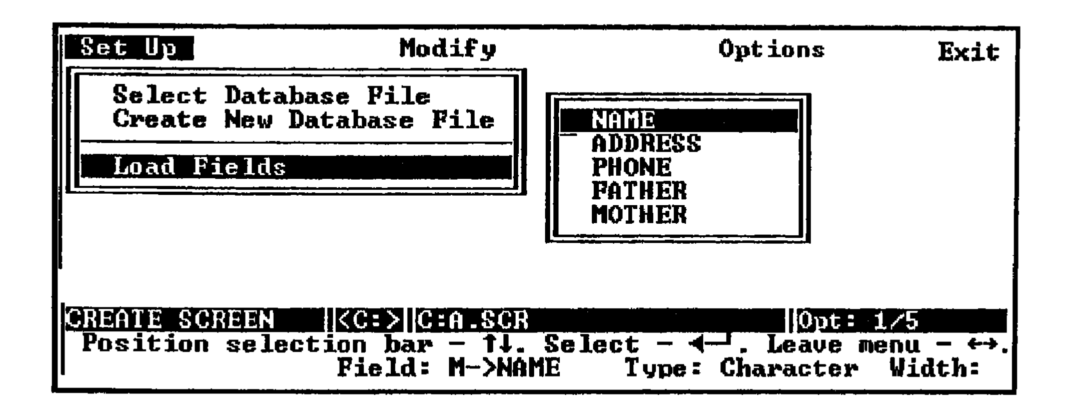
تحتوى قائمة تصميم الشاشة على القوائم الفرعية المبينة بالشكل (١٤ - ١)



شكل (١٤ - ١) قائمة تصميم الشاشة

١٤ - ١ - ١ التجهيـــز (Set Up)

وهى قائمة تتيح للمستخدم تحديد إسم ملف قاعدة البيانات المطلوب تصميم الشاشة له. كما تتيح له أيضا إختيار الحقول المطلوب إدخالها فى شاشة الإدخال. ويجب التفرقة هنا بين الحقول الفعلية الموجودة فى ملف قاعدة البيانات وبين الحقول المطلوب عرضها فى شاشة الإدخال. حيث أنها لاتكون بالضرورة نفس الحقول فى الحالتين. ففى بعض الأحيان قد لايكون مطلوبا ظهور بعض الحقول أمام القائم بإدخال البيانات. وذلك عندما تكون هذه البيانات سرية مثلا أو يتم إدخالها بصورة مجمعة أنظر الشكل (١٤ - ٢).



شكل (١٤ - ٢) قائمة التجهيز (Set Up)

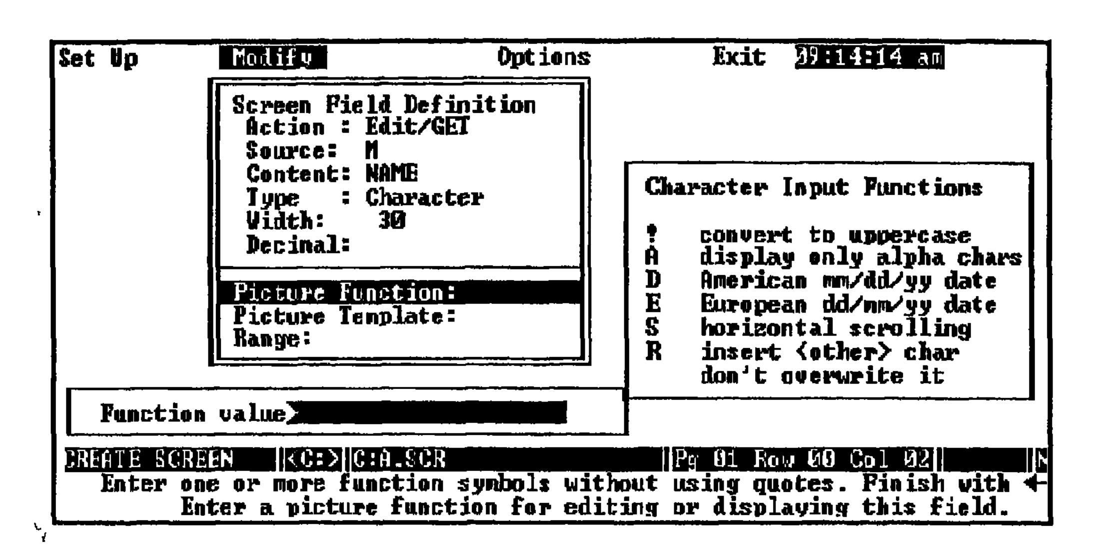
١٤ - ١ - ٢ التعـــديل (Modify)

وهى قائمة تسمح للمستخدم بإضافة حقول جديدة أو تعديل الحقول الموجودة. كما أنها تتيح له تعديل شكل المدخلات فى شاشة الإدخال وفى ملف قاعدة البيانات المفتوح انظر الشكل (١٤ - ٣).

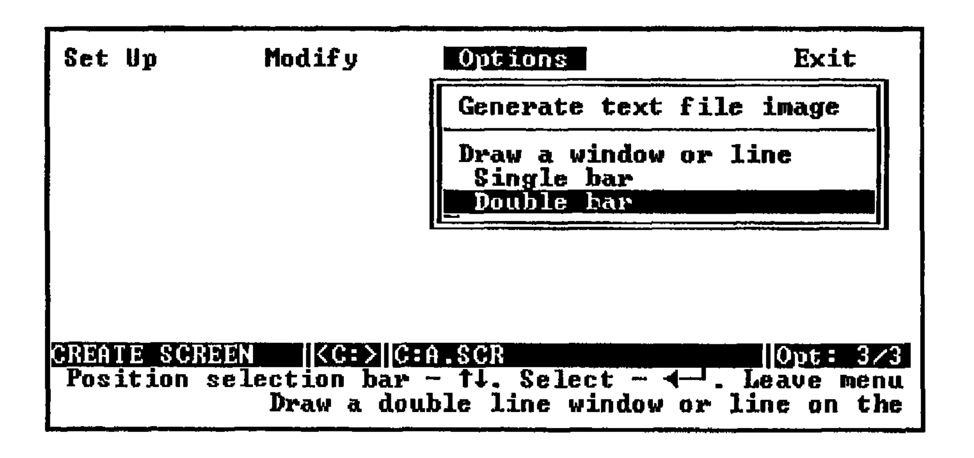
١٤ - ١ - ٣ الإختيارات (Options)

وهى قائمة تتيح للمستخدم الحصول على نسخة من ملف تعديل الشاشة مكتوبة بشفرة الآسكى (ASCII Code) حتى يمكن تعديلها بواسطة أى برنامج من برامج

معالجة الكلمات. كما أنها تتيح له استخدام الخطوط والمستطيلات فى تصميم شاشة الإدخال انظر الشكل (١٤ - ٤).



شكل (١٤ - ٣) قائمة التعديل (Modify)



شكل (١٤ - ٤) قائمة الإختيارات (Options)

١٤ - ١ - ٤ الخروج (Exit)

وهى قائمة تتيح للمستخدم الخروج من قوائم تعديل الشاشة بعد تخزين التعديلات أو دون تخزينها.

١٤ - ٢ خطوات تصميم شاشة الإدخال

لتصميم شاشة الإدخال يتم اتباع الخطوات التالية :

١ - يتم فتح قائمة الإنشاء (Create) واختيار (Format) كما سبق الإيضاح ثم كتابة إسم الملف المطلوب إنشاؤه فتظهر قوائم تصميم الشاشة كما سبق الإيضاح.

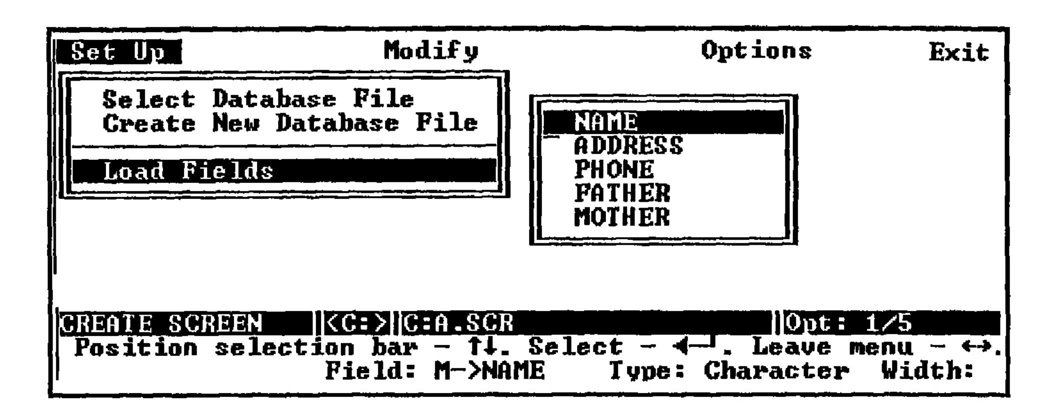
٢ - يكون المؤشر على قائمة التجهيز (Set Up) وبالتالى تكون قائمة التجهيز مفتوحة والعمود الضوئى (Highlight) موجودا على أول اختيار فيها وهو (Select Database File) فيتم الضغط على مفتاح الإدخال. وفى هذه الحالة يلاحظ ظهور قائمة فرعية تحتوى على جميع ملفات قواعد البيانات الموجودة على وحدة الأقراص المستخدمة. وهى كل الملفات التى تحتوى على الإمتداد (DBF.). ويظهر عمود ضوئى (Highlight) يتم عن طريقه اختيار الملف المطلوب فتحه ثم الضغط على مفتاح الإدخال.

٣ - يتم تحريك المؤشر باستخدام مفتاح السهم لأسفل(↓) إلى الإختيار (Load Fields) أى (تحميل الحقول) ثم الضغط على مفتاح الإدخال ، فيلاحظ ظهور قائمة فرعية تحتوى على أسماء الحقول الخاصة بملف قاعدة البيانات المفتوح ، فيتم اختيار الحقول المطلوب عرضها فى شاشة الإدخال ، ولايشترط عرض كل حقول ملف قاعدة البيانات كما سبق الإيضاح.

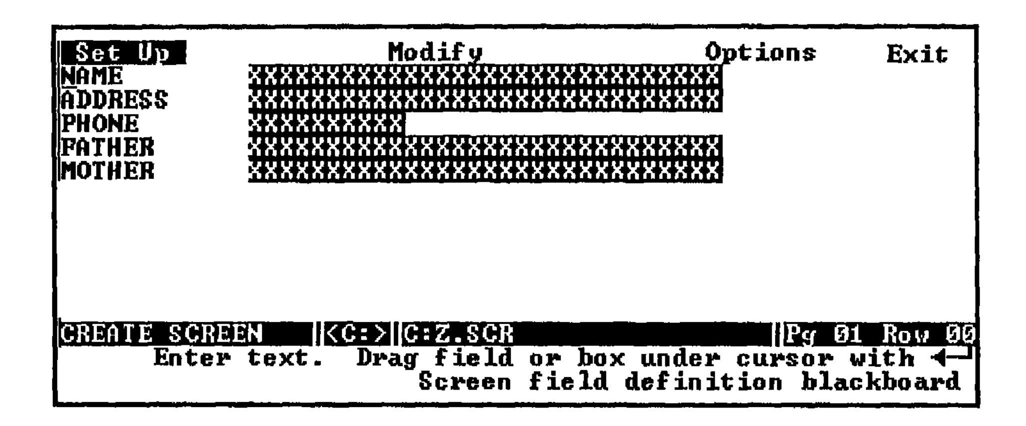
٤ - يتم اختيار الحقول عن طريق تحريك المؤشر إلى كل حقل مطلوب والضغط على مفتاح الإدخال فيلاحظ ظهور علامة (◀) أمام هذا الحقل فيتم الإنتقال إلى الحقل التالى والضغط على مفتاح الإدخال فتظهر العلامة (◀) أمامه. ويتم تكرار هذه العملية مع كل الحقول المطلوبة. ويمكن إلغاء العلامة أمام أى حقل بالضغط على مفتاح الإدخال مرة ثانية انظر الشكل (١٤ - ٥).

٥ - يتم الضغط على مفتاح السهم يمين (<---) أو مفتاح السهم شمال (--->) للخروج من قائمة الحقول. وهذا يعنى أن عملية اختيار الحقول قد انتهت فتختفى قائمة الحقول.

٦ - يتم الضغط على مفتاح (F10) فيلاحظ ظهور الحقول التى تم اختيارها أعلى الشاشة. ويلاحظ ظهور إسم كل حقل وأمامه العمود الضوئى (Highlight) الذى يمثل طول هذا الحقل. هذه الشاشة تسمى السبورة (Blackboard) لأنها تسمح للمستخدم برسم شاشة الإدخال بالطريقة والشكل الذى يريده كما سيتم الإيضاح انظر الشكل (١٤ - ٦).



شكل (١٤ - ٥) قائمة التجهيز



شكل (١٤ - ٦) شاشة السبورة (Blackboard)

ملاحظة

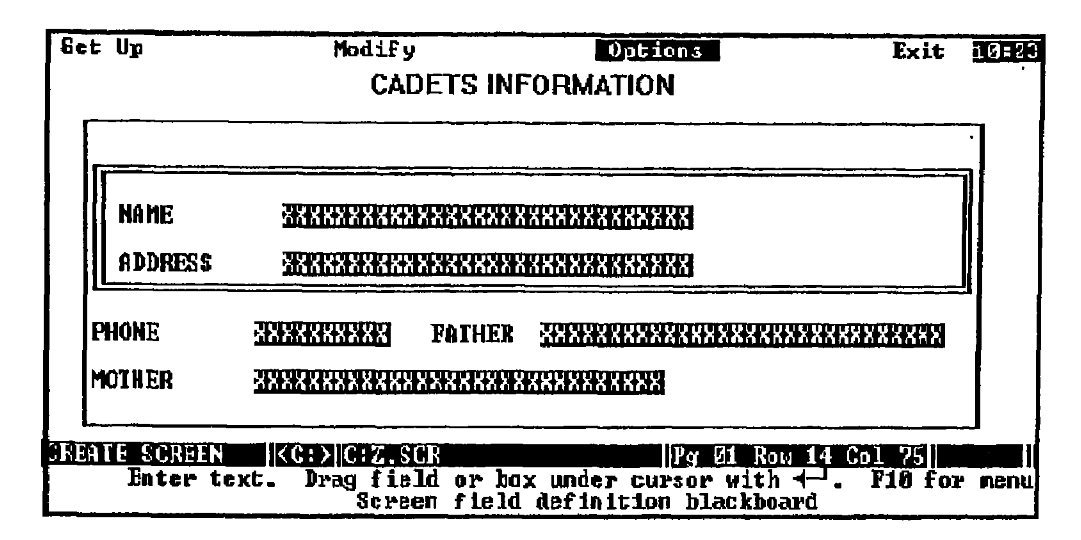
يلاحظ أن قائمة التجهيز(SetUp) تحتوى على الإختيار(Create New Database File) وهذا الإختيار يستخدم فى حالة إنشاء ملف قاعدة بيانات جديد عن طريق شاشة الإدخال. وفى هذه الحالة لايلزم إنشاء هيكل ملف قاعدة البيانات بالطريقة السابق شرحها.

١٤ - ٣ إستخدام السبورة (Blackboard)

كما سبق الإيضاح فإن الهدف من تصميم شاشة الإدخال هو الحصول على شاشة إدخال

للبيانات واضحة ومفهومة للمستخدم. كما أنها يجب أن تكون مقسمة وموزعة بطريقة تسهل للمستخدم إدخال البيانات بسهولة وبسرعة لأن الوقت المستهلك فى إدخال البيانات يكون غالبا أكبر من وقت الإسترجاع والتشغيل لهذه البيانات. ولذلك يكون من المهم تقليل هذا الوقت إلى الحد الأدنى.

وحتى يتم توضيح عملية تصميم الشاشة يمكن البدء بعرض الصورة النهائية للشاشة بعد عملية التصميم. انظر الشكل (١٤ - ٧).



شكل (١٤ - ٧) الصورة النهائية للشاشة بعد عملية التصميم

ومن هذا الشكل يلاحظ الآتى :

١ - تم إدخال عنوان للشاشة.
٢ - تم تنظيم أوضاع الحقول على الشاشة وترتيبها بطريقة منطقية مناسبة للقائم بإدخال البيانات.
٣ - تم تغيير أسماء الحقول مع الاحتفاظ بالأسماء الفعلية للحقول كما هى.
٤ - يلاحظ وجود مستطيلات حول البيانات بخطوط منفردة (Single) أو مزدوجة (Double).

ولكن كيف يتم تحويل شاشة الإدخال إلى هذه الصورة ؟ هذا ما سيتم إيضاحه فى الجزء التالى.

١٤ - ٤ مفاتيح التحكم فى الشاشة

تستخدم بعض مفاتيح لوحة المفاتيح (Keyboard) فى التحكم فى الحقول والكتابة الموجودة فى الشاشة. وهناك وظائف يمكن تنفيذها بعدة وسائل وبأكثر من مفتاح. وفيما يلى توضيح لهذه المفاتيح والوظيفة التى تؤديها مع ملاحظة أن وجود مفتاحين داخل القوس يعنى الضغط على المفتاحين فى نفس الوقت (Simultaneously).

- (Ins) أو (Ctrl-V): وتستخدم للتحويل بين وضع إضافة الحروف (Insert mode) ووضع الكتابة مع إلغاء الحروف السابقة. واستخدام هذه المفاتيح داخل الحقول يؤدى إلى توسيع عرض الحقل (Field Width) وذلك فى الشاشة فقط. بينما يظل عرضه ثابتا فى ملف قاعدة البيانات.
- (Ctrl-N): وتستخدم لإضافة سطر خال جديد بعد مكان المؤشر.
- (PgDn) أو (Ctrl-C): وتستخدم لتحريك الشاشة لأسفل صفحة كاملة (١٨ سطرا)
- (PgUp) أو (Ctrl-R): وتستخدم لتحريك الشاشة لأعلى صفحة كاملة (١٨ سطرا)
- مفاتيح الأسهم <--،--> ↓ ↑: وتستخدم لتحريك المؤشر خطوة واحدة فى الإتجاهات الموضحة بالأسهم.
- (Home) أو (Ctrl-A): وتستخدم لتحريك المؤشر إلى بداية الكلمة.
- (Ctrl -->) أو (Ctrl-B): وتستخدم لتحريك المؤشر إلى نهاية السطر.
- (End) أو (Ctrl-F): وتستخدم لتحريك المؤشر إلى بداية الكلمة التالية.
- (Enter) أو (Ctrl-M): وتستخدم لتحريك المؤشر إلى بداية السطر التالى. وفى حالة استخدام وضع الإضافة (Insert) فإن هذه المفاتيح تستخدم لإضافة سطر خال وإذا تم وضع المؤشر على العمود الضوئى (Highlight) الممثل لحقل من الحقول فإن هذه المفاتيح تستخدم لسحب

هذا العمود الضوئى وتحريكه إلى أى مكان. وعند وضع المؤشر على أى مستطيل(Box) فإن هذه المفاتيح تستخدم فى تغيير محيط هذا المستطيل بتكبيره أو تصغيره.

- (Ctrl <--) أو (Ctrl-Z) وتستخدم لتحريك المؤشر إلى بداية السطر.
- (Del) أو (Ctrl-G) وتستخدم لمسح الحرف عند مكان المؤشر. وإذا تم استخدامها فى حالة وجود المؤشر فى العمود الضوئى (Highlight) الخاص بالحقل فإنها تؤدى إلى تقليل عرض الحقل (Width) وذلك فى الشاشة فقط فى حين يظل عرض الحقل فى ملف قاعدة البيانات كما هو بدون تغيير.
- (Backspace) ويستخدم لمسح الحرف مكان المؤشر.
- (Ctrl-T) وتستخدم لمسح كل الحروف بدءا من مكان المؤشر حتى بداية الكلمة التالية.
- (Ctrl-Y) وتستخدم لمسح السطر مكان المؤشر.
- (Ctrl-U) وتستخدم لمسح العمود الضوئى (Highlight) الخاص بحقل معين مكان المؤشر. وكذلك تستخدم لمسح مستطيل (Box) يكون المؤشر واقعا على أى نقطة على محيطه.

١٤ - ٥ إضافة عنوان للشاشة

تبدأ عملية تصميم الشاشة بإضافة عدة سطور خالية فوق الشاشة لإدخال العنوان بها. ولتنفيذ ذلك يتم تحريك المؤشر إلى أول سطر فى الشاشة ثم بالضغط على مفتاحى (Ctrl-N) عدة مرات بعدد السطور المراد إدخالها يلاحظ تحرك جميع الحقول إلى أسفل مع إضافة عدة سطور خالية مكانها.

ويمكن تنفيذ هذه العملية أيضا عن طريق الضغط على مفتاح (Ins) للتحويل إلى وضع الإضافة (Insert) بدلا من وضع الكتابة مع الإلغاء (Overwrite) ثم الضغط على مفتاح الإدخال عدة مرات. يلاحظ فى هذه الحالة إضافة عدة سطور خالية مع تحرك الحقول إلى أسفل ثم يتم كتابة عنوان للشاشة فى هذه السطور.

تحــذير

عند الضغط على مفتاح (Ins) لتعديل الحالة يجب التأكد من عدم وقوف المؤشر على العمود الضوئى الخاص بأى حقل لأنه فى هذه الحالة سيؤدى إلى زيادة عرض العمود الضوئى (Highlight) بينما يؤدى الضغط على مفتاح (Del) إلى تقليل عرض العمود الضوئى.

ملاحـظة

يمكن إضافة سطور أخرى بين الحقول بنفس الطريقة باستخدام مفتاحى (Cntrl-N) أو مفتاح الإدخال فى وضع الإضافة (Insert).

١٤ - ٦ تحريـك الحقــول (Moving Fields)

لتحريك العمود الضوئى (Highlight) الخاص بأى حقل يتم اتباع الخطوات الآتية :

١ - يتم وضع المؤشر على أول العمود الضوئى (Highlight) ويلاحظ أن خط الرسائل (Message Line) الموجود أسفل الشاشة يوضح البيانات الخاصة بهذا الحقل مثل إسم الحقل ونوعه وعرضه.
٢ - يتم الضغط على مفتاح الإدخال.
٣ - يتم تحريك المؤشر إلى أى مكان على الشاشة يراد نقل الحقل اليه باستخدام مفاتيح الاسهم ثم يتم الضغط على مفتاح الإدخال مرة ثانية ويلاحظ إنتقال العمود الضوئى الممثل لهذا الحقل إلى المكان الجديد.
٤ - يتم كتابة عنوان جديد لهذا الحقل أمام هذا العمود الضوئى مع ملاحظة الضغط على مفتاح (Ins) لتحويل الحالة إلى وضع الإبدال أى الكتابة مع الحذف (Overwrite). ويمكن فى هذه الحالة كتابة أى عنوان واضح للحقل بدلا من الإسم السابق لأن الإسم فى هذه الحالة لايكون مقيدا بشروط معينة. فمثلا يمكن كتابة (Telephone Number) بدلا من (T_Num).
٥ - يتم حذف أسماء الحقول السابقة باستخدام مفتاح (Del) مع ملاحظة التحويل إلى حالة الكتابة مع الحذف (Overwrite) حتى لاتتحرك الأعمدة الضوئية الخاصة بالحقول.

١٤ - ٧ تعديل عرض الحقول (Field Width)

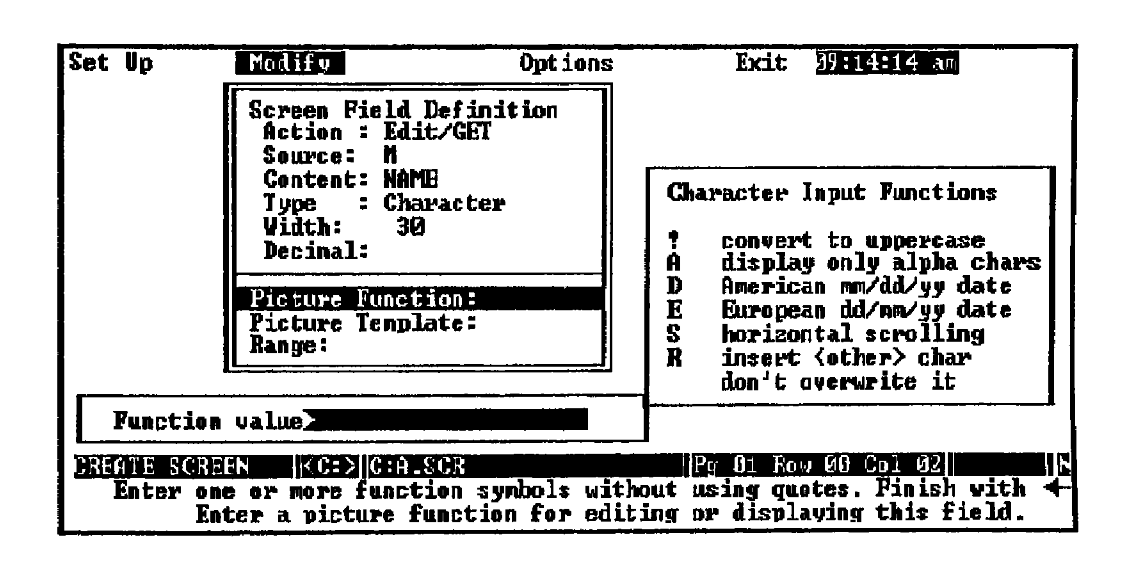
يمكن تعديل عرض الحقول فى الشاشة بطريقتين كالآتى :

١٤ - ٧ - ١ الطريقة الأولى

وهى كما سبق الإيضاح تكون عن طريق وضع المؤشر فى أى مكان داخل العمود الضوئى (Highlight) والضغط على مفتاح (Ins) لتوسيع الحقول أو الضغط على مفتاح (Del) لتقليل عرض الحقل. ويجب ملاحظة أن تغيير عرض الحقل فى هذه الحالة يتم بالنسبة للشاشة فقط ولكنه لا يؤثر فى عرض الحقل فى ملف قاعدة البيانات.

١٤ - ٧ - ٢ الطريقة الثانية

وهى عن طريق استخدام قائمة التعديل (Modify) ويتم ذلك كالآتى :
أنظر الشكل (١٤ - ٨)



شكل (١٤ - ٨)

١- يتم تحريك المؤشر إلى أول العمود الضوئى (Highlight) الخاص بالحقل المطلوب تعديل عرضه.
٢- يتم الضغط على مفتاح (F10) أو مفتاحى (Ctrl-Home) لفتح قائمة التعديل (Modify).
٣- يلاحظ عند فتح قائمة التعديل ظهور إسم الحقل الذى يقف عنده المؤشر بالإضافة إلى نوع الحقل وعرضه.
٤- يتم تحريك العمود الضوئى (Highlight) الخاص بقائمة التعديل (Modify) إلى الإختيار (Width) ثم الضغط على مفتاح الإدخال ، ويتم تعديل الرقم الموجود فى هذه الخانة إلى الرقم الجديد ثم الضغط على مفتاح الإدخال مرة ثانية.
٥- يتم الضغط على مفتاح (F10) مرة ثانية للخروج من قائمة التعديل. ويلاحظ ظهور السبورة (Blackboard) مرة ثانية مع ظهور العمود الضوئى الخاص بالحقل بالعرض الجديد الذى تم إدخاله.
٦- استخدام هذه الطريقة فى تعديل عرض العمود يؤدى إلى تعديل عرض العمود فى كل من الشاشة وملف قاعدة البيانات.

تحذير

عند إنقاص عرض حقول ملف قاعدة بيانات سبق إدخال بيانات به فإن ذلك يؤدى إلى اختفاء أى بيانات تزيد عن العرض الجديد. وذلك فى حالة الحقول الحرفية أما الحقول العددية فإن البرنامج يحتفظ بقيمتها ولكنه يظهر حروف (*) مكان أرقام العدد ليوضح أن العدد يزيد عن عرض الحقل المتاح.

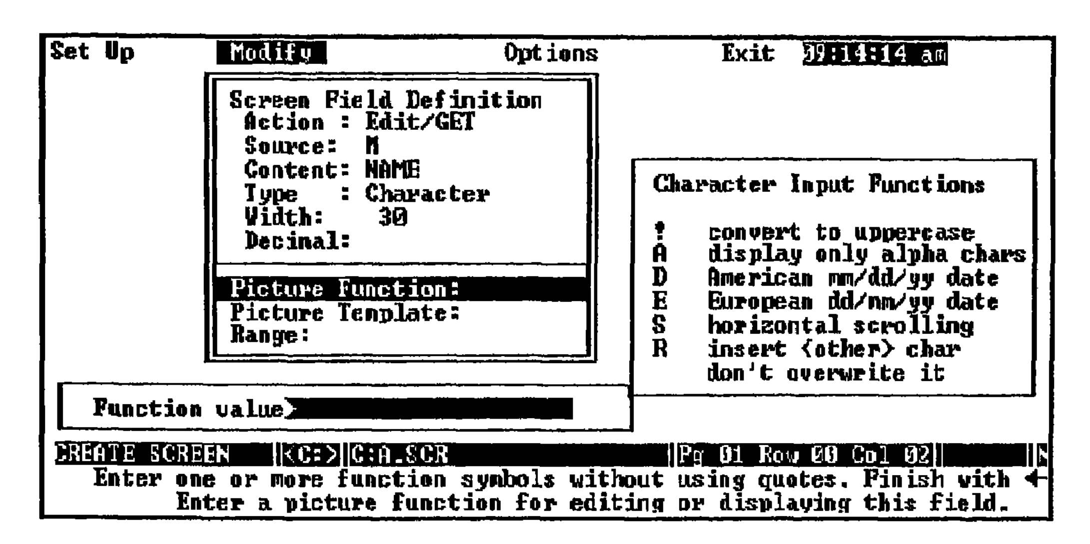
١٤ - ٨ إضافة حقول جديدة إلى شاشة الإدخال

يمكن إضافة حقول جديدة إلى شاشة الإدخال مع إضافتها إلى قاعدة البيانات فى نفس الوقت وذلك كالآتى :

انظر الشكل (١٤ - ٩)

١- يتم تحريك المؤشر إلى أى مكان داخل شاشة الإدخال يراد وضع الحقل فيه ثم يتم الضغط على مفتاح (F10) فيلاحظ فتح قائمة التعديل (Modify) ويلاحظ أن

القائمة تظهر خالية أى لا تحتوى على أى بيانات مكان إسم الحقل والنوع والعرض. وذلك لأن مؤشر شاشة الإدخال كان موجودا فى مكان خال وليس فى حقل معين.



شكل (١٤ - ٩)

٢- يتم تحريك العمود الضوئى (Highlight) إلى (Content) ثم يتم كتابة إسم الحقل الجديد المطلوب إضافته وليكن مثلا (B_date) أى تاريخ الميلاد. مع ملاحظة أن الإسم هنا يخضع للشروط العامة لأسماء الحقول السابق ذكرها.

٣ - يتم ملء البيانات الأخرى الخاصة بالنوع (Type) وعرض الحقل (Width).

٤ - يتم الضغط على مفتاح (F10) مرة ثانية فيتم العودة إلى شاشة الإدخال ويلاحظ ظهور العمود الضوئى الخاص بهذا الحقل.

٥ - يتم كتابة عنوان لهذا الحقل أمام العمود الضوئى فى شاشة الإدخال وليكن مثلا (Birth Date) مع ملاحظة أن العنوان هنا لايشترط أن يكون هو نفس إسم الحقل ولكن يفضل أن يكون عنوانا واضحا للشخص القائم بعملية إدخال البيانات كما سبق الإيضاح.

١٤ - ٩ مسح حقول من شاشة الإدخال

قد يريد المستخدم فى بعض الأحيان مسح بعض الحقول التى يراها غير مطلوبة فى شاشة الإدخال. وقد يريد أيضا مسح بعض الحقول من الشاشة ومن ملف قاعدة البيانات معا. ويتم ذلك باتباع الخطوات التالية :

١ - يتم وضع المؤشر على العمود الضوئى الخاص بالحقل المطلوب مسحه.
٢ - يتم الضغط على مفتاحى (Ctrl-U).
٣ - يظهر سؤال على الشاشة عما إذا كان المطلوب مسح الحقل من الشاشة فقط أم من الشاشة وملف قاعدة البيانات فى نفس الوقت.
٤ - يتم كتابة (N) عندما يراد المسح من الشاشة فقط ويتم كتابة (Y) عندما يراد المسح من الشاشة وملف قاعدة البيانات معا.
٥ - يتم مسح عنوان الحقل من الشاشة باستخدام المفتاح (Del) كما سبق الإيضاح.

تحـذيـر

عند مسح حقل من شاشة الإدخال والملف فى نفس الوقت فإن أى بيانات سبق تخزينها فى هذا الحقل سوف تفقد.

١٤ - ١٠ تعديل خصائص الحقل على الشاشة

يتيح برنامج (+ DBase III) للمستخدم التحكم فى خصائص الحقل على الشاشة دون ان يؤثر هذا التحكم على الحقل الفعلى فى ملف قاعدة البيانات. حيث يمكن للمستخدم مثلا تحديد مدى معين (Range) للمدخلات. كما يمكنه أيضا أن يسمح للقائم بإدخال البيانات بإدخال البيان أو لايسمح له بذلك حسب الحاجة. ويتم ذلك باستخدام بعض الإختيارات الموجودة فى قائمة التعديل وهى الإختيارات الآتية :

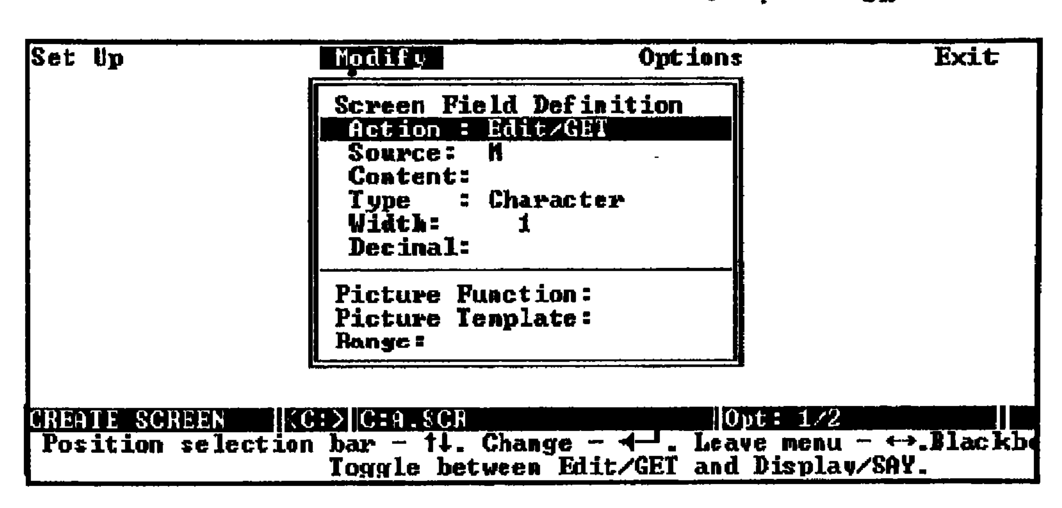
Action , Picture Function , Picture Template , Range.

ويتم شرح هذه الإختيارات فى الأجزاء التالية :

١٤ - ١٠ - ١ الإختيار (Action) أو الفعل

وهو يعنى الفعل المسموح به للقائم بإدخال البيانات فى هذا الحقل سواء كان مجرد رؤية بيانات الحقل دون القدرة على تغييرها أو تغيير هذه البيانات. ويلاحظ أن الوضع المبدئى لهذا الإختيار يكون (Edit \ Get) ومعناه أنه يمكن رؤية بيانات هذا الحقل وتعديلها. ولكى يتم تعديل هذا الإختيار يتم إتباع الخطوات التالية. أنظر شكل (١٤ - ١٠).

١- من السبورة (Blackboard) يتم تحريك المؤشر حتى يصل إلى العمود الضوئى (Highlight) الممثل للحقل المطلوب تعديله.
٢- يتم الضغط على مفتاح (F10) فتظهر قائمة التعديل (Modify).
٣- يلاحظ ظهور بيانات هذا الحقل فى قائمة التعديل مثل إسم الحقل ونوعه وعرضه.
٤- يتم تحريك العمود الضوئى (Highlight) إلى الإختيار (Action) ويلاحظ أن الوضع المبدئى يكون مكتوبا وهو (Edit/Get) ومعناه أنه يمكن رؤية بيانات هذا الحقل وتعديلها.
٥- يتم الضغط على مفتاح الإدخال فيلاحظ تحول الإختيار إلى (Display/Say) وهو يعنى أن بيانات الحقل للعرض فقط ولايمكن تعديل البيانات أو مسحها.
٦- يتم الضغط على مفتاح (F10) مرة ثانية للعودة إلى السبورة (Blackboard) ويلاحظ إختفاء العمود الضوئى (Highlight) الخاص بهذا الحقل. وهذا يعنى أن هذا الحقل سوف يعرض البيانات الموجودة فقط ولكنه لن يسمح بتعديلها عن طريق شاشة الإدخال.



شكل (١٤ - ١٠)

١٤ - ١٠ - ٢ الاختيار (Picture Function) أو دالة الصورة

وهذا الاختيار يسمح للمستخدم بعمل تحويل للمدخلات قبل دخولها إلى ملف قاعدة البيانات. فمثلا يمكن تحويل المدخلات الحرفية إلى حروف كبيرة

(Uppercase) ، بحيث تتحول دائما إلى حروف كبيرة بصرف النظر عن الشكل الذى أدخلت به. فيمكن فى هذه الحالة أن يقوم القائم بإدخال البيانات بإدخال المدخلات الحرفية بحروف كبيرة (Uppercase) أو صغيرة (Lowercase) مع دخولها فى جميع الأحوال بحروف كبيرة.

ولتنفيذ ذلك يتم اتباع الخطوات التالية :

١- يتم تحريك المؤشر على السبورة (Blackboard) حتى يصل إلى العمود الضوئى (Highlight) الخاص بالحقل المطلوب تعديله.
٢- يتم الضغط على مفتاح (F10) لإظهار قائمة التعديل (Modify). ويلاحظ ظهور بيانات الحقل مثل الإسم والنوع والعرض.
٣- يتم تحريك العمود الضوئى بقائمة التعديل حتى يصل إلى الاختيار (Picture Function) ثم الضغط على مفتاح الإدخال.
٤ - يلاحظ ظهور رسالة أسفل الشاشة كالآتى :

Function Value

وأمامها عمود ضوئى لادخال القيمة المطلوبة. كما تظهر قائمة على يمين الشاشة توضح الاختيارات المختلفة لهذا الحقل والتى سيتم شرحها فى الجزء التالى.

٥- يتم كتابة الحرف (!) فى العمود الضوئى لتحويل الحروف إلى حروف كبيرة (Uppercase) والضغط على مفتاح الإدخال.
٦- يتم الضغط على مفتاح (F10) للعودة الى السبورة.

انظر الشكل (١٤ - ١١)

ويلاحظ من قائمة اختيارات دالة الصورة (Picture Function) أن هناك خمسة اختيارات وهى (!) ، (A) ، (R) ، (D) ، (E) ، ويتم شرحها فى الجدول التالى :

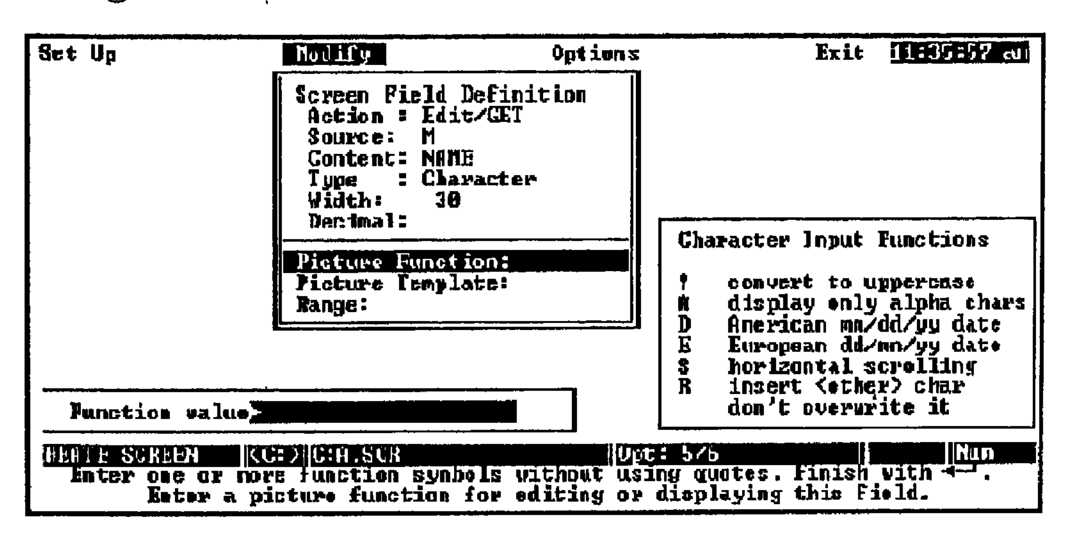
!	وهو يؤدى إلى تحويل كل الحروف التى يتم إدخالها إلى حروف كبيرة.
A	وهو يحول الحروف إلى حروف كبيرة (Uppercase) مع إدخال الحروف الهجائية فقط (Alphabet) وعدم قبول أى حروف خاصة (Special Characters) أو أعداد.

D وهو يؤدى إلى إدخال التاريخ على النظام الأمريكى شهر / يوم / سنة (mm/dd/yy).

E وهو يؤدى إلى إدخال التاريخ على النظام الأوربى يوم / شهر / سنة (dd/mm/yy).

R وهو يعمل على الحقول الحرفية مع تحديد شكل معين لها(Template). حيث يمكن وضع علامات أو حروف خاصة بحيث تعمل كفواصل بين الحروف ولايمكن الكتابة فوقها. فمثلا إذا كانت هناك مدخلات يراد إدخالها مع وجود فواصل خالية بين الحروف يتم كتابة الشكل الآتى مثلا :

@R A A A A A A A



شكل (١٤ - ١١)

١٤ - ١٠ - ٣ الاختيار (Picture Template) أو هيكل الصورة

وهذا الاختيار يسمح للمستخدم بتحديد نوع المدخلات المسموح بها فى هذا الحقل. بمعنى أن الحقل لايقبل أى مدخلات تخالف النوع الذى يتم تحديده من خلال هذا الأختيار. كما يسمح للمستخدم أيضا بتحديد شكل معين (Format) لهذه المدخلات. فمثلا يمكن تحديد شكل رقم التليفون كالآتى :

(XXX)XXX-XXXX

وذلك بالنسبة للدول التى تستخدم هذا الشكل من الأرقام. وهناك عدة اختيارات تظهر أيضا عند استخدام هذا الاختيار يتم توضيحها فى الجدول التالى :

A	وهى تعنى إدخال الحروف الكبيرة (Uppercase). وفى هذه الحالة لايقبل الحقل أى حروف صغيرة (Lowercase). وهذا يختلف عن الاختيار (!) الذى يحول الحروف الصغيرة التى يتم إدخالها إلى حروف كبيرة.
L	وهو يسمح فقط بإدخال المدخلات المنطقية التى تشمل (T, F, Y, N) ولا يقبل أى حرف آخر.
X	وهو يسمح بإدخال الحروف التى يمكن أن تشتمل على أرقام أو حروف خاصة (Special Characters).
#	وهو يسمح بإدخال الأعداد التى يمكن أن تشتمل على مسافات وعلامات الجمع (+) وعلامات الطرح (-).
9	وهو يسمح بإدخال الأعداد فقط.
!	وهو يحول الحروف إلى حروف كبيرة (Uppercase).
Other	وهو أى شكل يتم تحديده مثل ###.###(###).

١٤ - ١٠ - ٤ الاختيار (Range) أو المدى

وهو الاختيار الذى يسمح للمستخدم بتحديد حد أدنى وحد أقصى للمدخلات. وفى هذه الحالة فإن القائم بإدخال البيانات لايمكنه إدخال أى قيم خارج هذا المدى ، لأنها لن تقبل فى هذا الحقل. والقائمة الخاصة بهذا الاختيار تحتوى على الآتى :

Lower Limit	وتعنى أقل قيمة عددية مسموح بها.
Upper Limit	وتعنى أكبر قيمة عددية مسموح بها.

١٤ - ١٠ - ٥ إضافة الرسومات إلى شاشة الإدخال

يتيح البرنامج للمستخدم رسم مستطيلات حول بعض أو كل الحقول تفيد فى توضيح شاشة الإدخال أو توضيح أهمية بعض الحقول ، بالإضافة إلى جعل هيئة

الشاشة مقبولة ومريحة للقائم بإدخال البيانات. ويتم ذلك عن طريق اتباع الخطوات التالية :

١- يتم الضغط على مفتاح (F10) للرجوع إلى قائمة التعديل (Modify).
٢- يتم تحريك المؤشر العلوى إلى قائمة (Options) التى تسمح برسم نافذة (Window) بخطوط مفردة (Single) أو خطوط مزدوجة (Double). ثم يتم الرجوع إلى السبورة (Blackboard).
٣- يتم وضع المؤشر على الركن العلوى من اليسار (Upper Left) للمستطيل المطلوب رسمه ، والضغط على مفتاح الإدخال.
٤- يتم وضع المؤشر على الركن السفلى من اليمين (Lower Right) للمستطيل المطلوب رسمه ، والضغط على مفتاح الإدخال . فيلاحظ رسم مستطيل فى هذا المكان.

ملاحظات

١- المستطيلات أو الخطوط التى يتم رسمها على شاشة الإدخال لاتظهر عند طباعة هذه الشاشة على الطابعة كخطوط. بل تظهر غالبا كحروف هجائية(Alphabet).
٢- إذا أريد توسيع أو تصغير أى مستطيل يتم وضع المؤشر على أى جانب من المستطيل أو على ركنه والضغط على مفتاح الإدخال. ثم يتم نقل المؤشر إلى أى نقطة أخرى والضغط على مفتاح الإدخال مرة ثانية. ويلاحظ تعديل محيط المستطيل.
٣- عندما يراد مسح أى مستطيل يتم وضع المؤشر على أى نقطة على محيطه والضغط على مفتاحى (Ctrl-U) فيلاحظ إختفاء المستطيل.

١٤ - ١٠ - ٦ طباعة شاشة الإدخال

يمكن طباعة شاشة الإدخال عن طريق الضغط على مفتاح (PrtSc). كما يمكن تخزين الشاشة كملف نص (Text File) وطباعتها فى أى وقت باستخدام أوامر نظام التشغيل. ولتنفيذ ذلك يتم اتباع الخطوات التالية :

١ - يتم الضغط على مفتاح (F10) للرجوع إلى قائمة رسم الشاشة.
٢ - يتم فتح قائمة الاختيارات (Options Menu).
٣ - يتم اختيار (Generate Text File Image).

١٤ - ١٠ - ٧ تخزين شاشة الإدخال

يتم تخزين شاشة الإدخال حتى يتم استخدامها بعد ذلك فى إدخال البيانات إلى ملف قاعدة البيانات. ويتم تنفيذ ذلك باتباع الخطوات التالية :

١ - يتم الضغط على مفتاح (F10) للرجوع الى قائمة رسم الشاشة.
٢ - يتم فتح قائمة الخروج (Exit) ، واختيار الأمر (Save).


Converted by Tiff Combine - (no stamps are applied by registered version)

الفصل الخامس عشر

تحديث السجلات

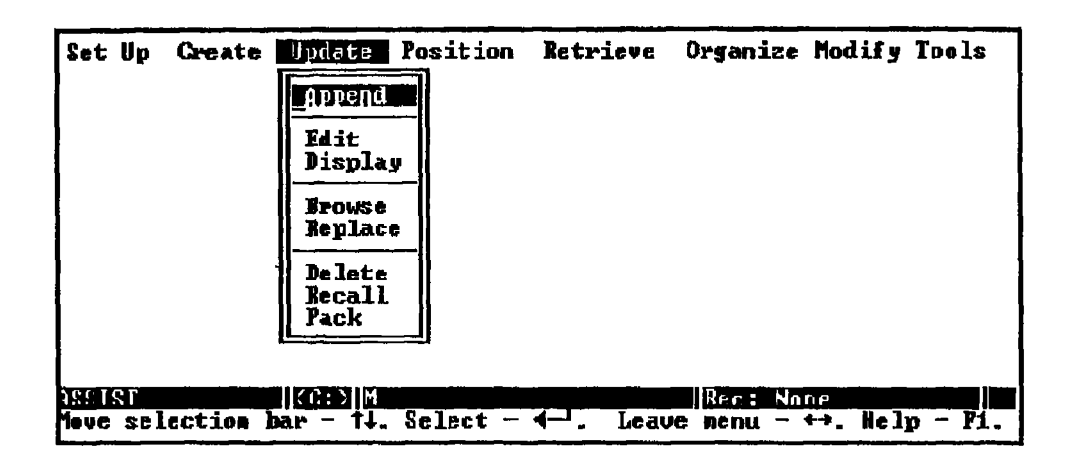
(Updating Records)


Converted by Tiff Combine - (no stamps are applied by registered version)

يوضح هذا الفصل عمليات عرض البيانات أو تعديلها أو إضافة بيانات جديدة أو حذف بيانات أو إضافة سجلات كاملة أو حذف سجلات كاملة وذلك باستخدام قائمة التحديث (Update) وهى إحدى القوائم الرئيسية الثمانية للبرنامج.

وتحتوى قائمة التحديث (Update) على الإختيارات التالية :

انظر الشكل (١٥ - ١)



شكل (١٥ - ١) إختيارات قائمة التحديث (Update)

١ - الإضافة (Append)

وهو يساعد على إضافة سجلات إلى نهاية ملف قاعدة البيانات كما يساعد على تصحيح السجلات التى سبق إدخالها.

٢ - التصحيح (Edit)

وهو يساعد على عرض وتصحيح السجلات الموجودة سجلا تلو الآخر.

٣ - العرض (Display)

وهو يساعد على عرض حتى ١٥ سجلا فى المرة الواحدة.

٤ - العرض مع التصحيح (Browse)

وهو يساعد على عرض وتصحيح وإضافة سجلات. وهو يؤدى إلى عرض حتى ١٧ سجلا على الشاشة فى المرة الواحدة.

٥ - الإستبدال (Replace)

وهو يساعد على استبدال محتويات حقل معين بمدخلات جديدة فى سجل أو عدة سجلات.

٦ - المسح (Delete)

وهو يساعد على تحديد السجلات المطلوب مسحها ولكنه لايقوم بمسحها فعليا.

٧ - الإستعادة (Recall)

وهو يساعد على إستعادة السجلات التى سبق إعدادها للمسح حتى لايتم مسحها بواسطة الاختيار (Pack).

٨ - المسح النهائى (Pack)

وهو يساعد على مسح السجلات التى سبق تحديدها بواسطة الإختيار (Delete).

وفى الأجزاء التالية يتم دراسة هذه الإختيارات بالتفصيل.

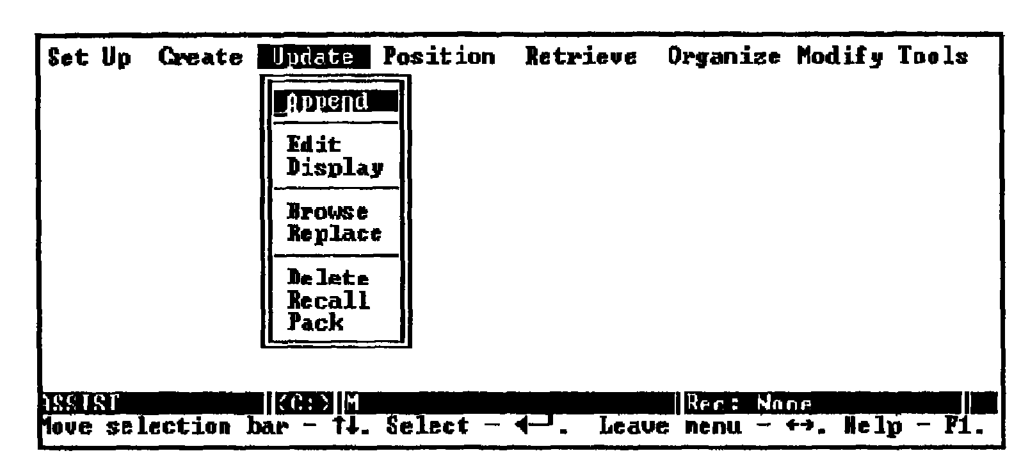
١٥ - ١ الإضافة (Append)

أنظر الشكل (١٥ - ٢)

ويستخدم هذا الإختيار عندما يراد إضافة سجلات جديدة بعد آخر سجل سبق تخزينه فى ملف قاعدة البيانات. ولتنفيذ ذلك يتم اتباع الخطوات التالية :

١ - يتم فتح قائمة التحديث (Update) واختيار (Edit).

٢ - يلاحظ فتح شاشة الإدخال إذا كان قد سبق اختيار شاشة إدخال معينة أو تظهر الشاشة المبدئية (Default) الخاصة ببرنامج (+DBaseIII).
٣ - يلاحظ وقوف مؤشر صغير على الحقل الأول يتم عن طريقه كتابة بيانات هذا الحقل.
٤ - يلاحظ أن المؤشر ينتقل إلى الحقل التالى فى حالة إمتلاء الحقل بالبيانات. وفى حالة عدم إمتلائه يلزم الضغط على مفتاح الإدخال حتى ينتقل المؤشر إلى الحقل التالى.



شكل (١٥ - ٢) الإضافة

وعلاوة على إضافة سجلات جديدة فإن هذا الإختيار يتيح عرض السجلات السابقة المخزنة فى الملف وتعديلها أيضا. ولعرض السجلات المخزنة فى الملف يتم استخدام مفتاحى (PgUp) ، (PgDn). حيث أن مفتاح (PgUp) يعرض السجلات التالية للسجل الحالى ومفتاح (PgDn) يعرض السجلات السابقة للسجل الحالى. كما أن مفتاحى (↓ ↑) ينقلان المؤشر إلى الحقول المختلفة داخل السجل. أما مفتاحا (<-- ، -->) فيستخدمان لتحريك المؤشر داخل الحقل حرفا واحدا فى كل مرة.

١٥ - ٢ التصحيح (Edit)

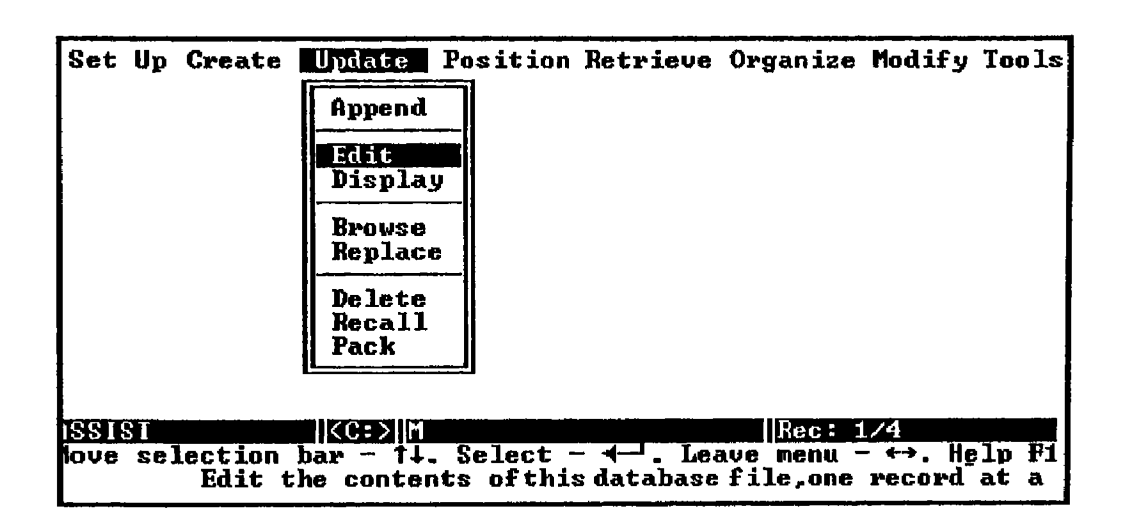
ويستخدم هذا الإختيار عندما يراد تعديل بيانات سجل معين فى ملف قاعدة البيانات ، انظر الشكل (١٥ - ٣). ولتنفيذ ذلك يتم اتباع الخطوات التالية :

١ - يتم فتح قائمة التحديث (Update) واختيار (Edit).
٢ - يلاحظ فتح شاشة الإدخال إذا كان قد سبق اختيار شاشة إدخال معينة أو تظهر

الشاشة المبدئية (Default) الخاصة ببرنامج (Dbase III+). وذلك بالنسبة للسجل الذى يكون قد سبق تحديده بواسطة الأمر (Locate) كما سيتم الإيضاح.

٣ - يلاحظ وقوف المؤشر فى الحقل الأول ويتم تعديل البيانات الموجودة به. وفى هذه الحالة يجب التأكد أن البرنامج فى حالة الكتابة مع إزالة الحروف السابقة (Overwrite Mode). وذلك بالتأكد من عدم ظهور كلمة (Ins) على عمود الحالة (Status Bar).

٤ - بعد إنتهاء التعديل فى السجل يتم الضغط على مفتاحى (Ctrl-End) لتخزينه.



شكل (١٥ - ٣) التصحيح

ملاحظة

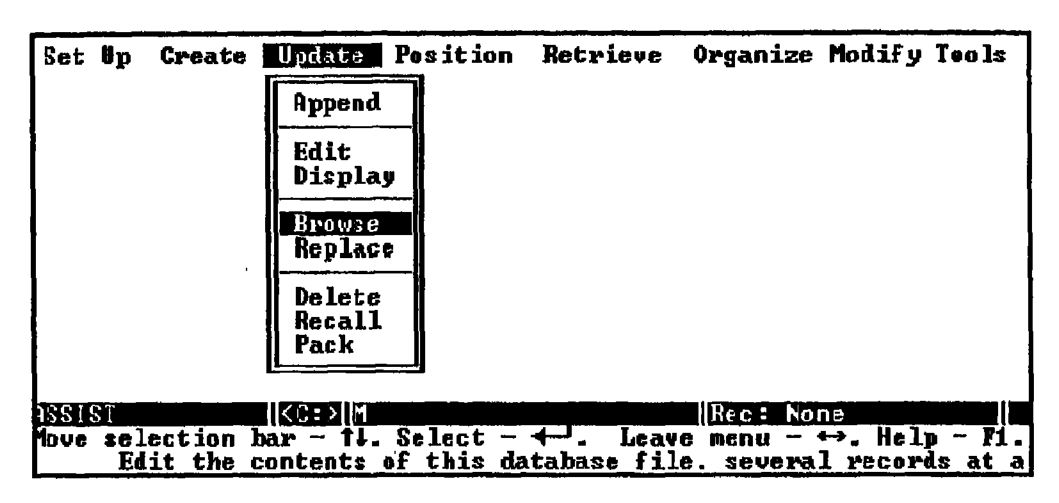
فى حالة عمل تعديلات فى السجل وعدم الرغبة فى تخزين هذه التعديلات يتم الضغط على مفتاح الهروب (Esc) بدلا من مفتاحى (Ctrl-End).

١٥ - ٣ العرض (Display)

ويستخدم هذا الاختيار لعرض حتى ١٥ سجلا فى المرة الواحدة. وهذا الاختيار نادرا مايستخدم من قائمة التحديث (Update) لأن هناك أمرا آخر فى قائمة الإسترجاع (Retrieve) يؤدى نفس الغرض.

١٥ - ٤ العرض مع التصحيح (Browse)

ويستخدم هذا الإختيار عندما يراد عرض السجلات في صورة جدول (Table) مع إمكانية التعديل فيها وكذلك إضافة سجلات جديدة ، أنظر الشكل (١٥ - ٤).



شكل (١٥ - ٤) العرض مع التصحيح

وهذا الإختيار يسمح بعرض حتى ١٧ سجلا في المرة الواحدة وكل سجل يظهر على سطر. وفي حالة إحتواء السجل على عدد كبير من الحقول تظهر الحقول الأولى من اليسار بقدر إتساع الشاشة.

ويمكن عرض جميع الحقول باستخدام مفتاحي الأسهم (<-,->) مع مفتاح (Ctrl) لإزاحة الشاشة إلى اليسار أو إلى اليمين على الترتيب.

ولتنفيذ عملية العرض باستخدام الإختيار (Browse) يتم اتباع الخطوات التالية :

١- يتم فتح قائمة التحديث (Update) واختيار (Browse).

٢- يلاحظ ظهور السجلات مع وقوف العمود الضوئي (Highlight) على أحد السجلات. ويكون هذا السجل هو السجل الذى سبق تحديده باستخدام الأمر (Locate) كما سيتم الإيضاح فيما بعد. فإذا لم يكن قد سبق تحديده يقف العمود الضوئي على أول سجل في ملف قاعدة البيانات.

٣- يلاحظ أن عمود الحالة (Status Bar) يوضح الإختيار الحالى (Browse) ووحدة الأقراص المستخدمة وإسم الملف المستخدم ورقم السجل الحالى الذى يقف عليه العمود الضوئى وعدد سجلات الملف.

٤- يلاحظ وجود مستطيل أعلى الشاشة يوضح المفاتيح التى يمكن استخدامها فى التحكم فى المؤشر. فإذا أريد إخفاء هذا المستطيل من الشاشة يستخدم المفتاح(F1). وإذا أريد إظهاره مرة ثانية يتم الضغط على مفتاح (F1) مرة ثانية أيضا.

٥- يلاحظ ظهور مؤشر صغير داخل أول حقل فى السجل الذى يقف عنده العمود الضوئى. وهذا المؤشر يستخدم فى إجراء التعديل المطلوب فى هذا الحقل. كما يمكن نقل هذا المؤشر من الحقل إلى الحقل الذى يليه باستخدام مفتاح (End). كما يمكن نقل المؤشر إلى الحقل الذى يسبقه باستخدام مفتاح (Home).

٦- لإظهار الحقول المختفية يتم إزاحة الشاشة إلى اليسار (Scroll) باستخدام مفتاحى (Ctrl,-->). كما يمكن تحريك الشاشة إلى اليمين باستخدام مفتاحى (Ctrl, <--).

٧ - عند الضغط على مفتاح (F10) أو مفتاحى (Ctrl-Home) يلاحظ ظهور قائمة جديدة أعلى الشاشة تتيح للمستخدم بعض الإختيارات التي تفيد فى التحكم فى الحقول والسجلات المخزنة فى ملف قاعدة البيانات. هذه الإختيارات تكون كالآتى :

أ - القاع (Bottom).
ب - القمة (Top).
جـ - القفل (Lock).
د - رقم السجل (Record No.).
هـ - التجمد (Freeze).
و - البحث (Seek).

ويتم شرح خصائص كل اختيار من هذه الإختيارات فى الأجزاء التالية :

١٥ - ٤ - ١ القاع (Bottom)

وهو يؤدى إلى ظهور آخر سجل فى ملف قاعدة البيانات مع وقوف العمود الضوئى (Higlight) عليه. ويلاحظ فى هذه الحالة ظهور رقم هذا السجل فى عمود الحالة (Status Bar). كما أن هذا الإختيار يتيح للمستخدم إضافة سجلات جديدة بعد آخر سجل فى الملف. ولتنفيذ ذلك يتم الضغط على مفتاح السهم لأسفل (↓)

فيلاحظ ظهور السؤال التالى :

Add new records ? (Y/N)

فيتم كتابة (Y) لإضافة سجل جديد.

١٥ - ٤ - ٢ القمة (Top)

وهو يؤدى إلى ظهور أول سجل فى ملف قاعدة البيانات مع وقوف العمود الضوئى (Highlight) عليه.

١٥ - ٤ - ٣ القفل (Lock)

وهو يؤدى إلى تثبيت الحقل الموجود فى أقصى يسار الشاشة مع إمكانية تحريك باقى الحقول بالنسبة لهذا الحقل. وعند استخدام هذا الاختيار يظهر سؤال على الشاشة عن رقم العمود المراد تثبيت جميع الحقول الموجودة إبتداء منه ويساره.

١٥ - ٤ - ٤ رقم السجل (Record No.)

وهو يؤدى إلى الوصول إلى سجل معين عن طريق رقم هذا السجل. وعند استخدام هذا الاختيار يظهر سؤال عن رقم السجل المطلوب. وفى هذه الحالة يقف العمود الضوئى (Highlight) على هذا السجل.

١٥ - ٤ - ٥ التجمد (Freeze)

وهو يؤدى إلى حماية كل الحقول من التعديل ماعدا الحقل المطلوب تعديله. وفى هذه الحالة يتم تحديد الحقل المطلوب تعديله حتى يتم حماية باقى الحقول والسماح بتعديل هذا الحقل فقط.

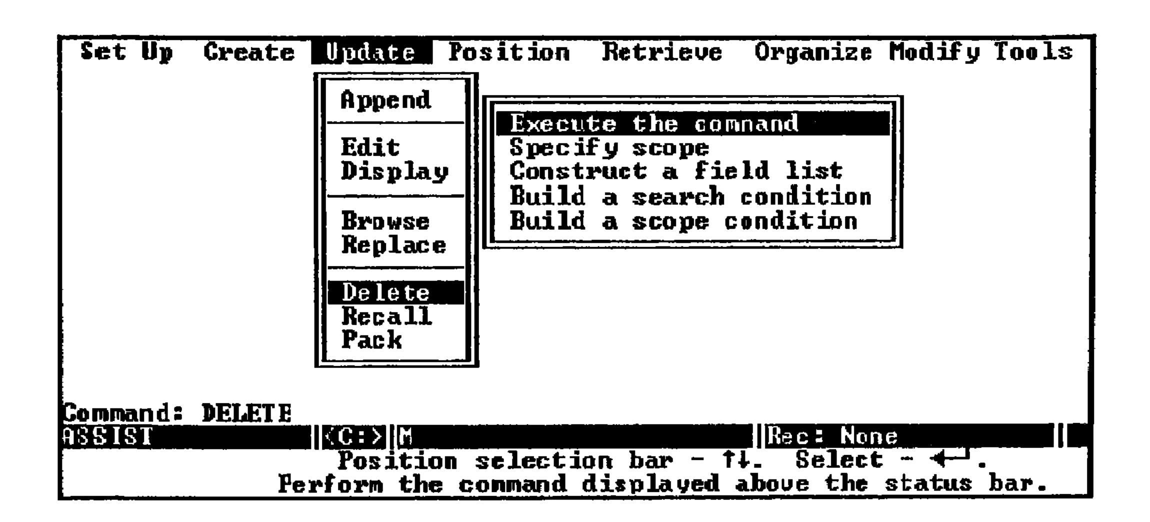
١٥ - ٤ - ٦ البحث (Seek)

وهو يؤدى إلى البحث عن سجل معين يحتوى على مجموعة حروف (Character String) أو قيمة عددية. وفى هذه الحالة يتم كتابة هذه الحروف

أو القيمة العددية حتى يبحث البرنامج عنها فى جميع السجلات. وهذا الاختيار لايظهر إلا فى حالة إنشاء فهرس (Index) للملف.

١٥ - ٥ المسح (Delete)

يستخدم هذا الاختيار لمسح سجل أو سجلات معينة من ملف قاعدة البيانات. وهو فى الواقع لايمسح السجلات ولكنه يقوم بتحديد هذه السجلات حتى يتم مسحها نهائيا باستخدام الاختيار (Pack). ولتنفيذ هذه العملية يتم اتباع الخطوات التالية أنظر شكل (١٥ - ٥).



شكل (١٥ - ٥) قائمة المسح

١- يتم فتح قائمة التحديث (Update).
٢- يتم تحريك العمود الضوئى (Highlight) حتى يصل إلى الاختيار (Delete) ثم الضغط على مفتاح الإدخال.
٣- يلاحظ فتح قائمة جديدة خاصة بتحديد شروط البحث (Search Conditions) ومدى البحث (Search Scope).
٤- يتم تحريك العمود الضوئى الخاص بقائمة البحث حتى يصل إلى (Build a Search Condition) والضغط على مفتاح الإدخال. يلاحظ ظهور قائمة بأسماء الحقول الموجودة فى الملف والتى منها يتم اختيار الحقل أو الحقول التى يتم إدخالها فى شروط البحث (Search Conditions).

٥- يلاحظ ظهور قائمة بمعاملات المقارنة التى يتم استخدامها فى شروط البحث فيتم اختيار المعامل المطلوب.

٦- يلاحظ خلال ذلك ظهور الأمر الذي يقوم بتنفيذ هذه العمليات فى سطر الأوامر (Command Line) أسفل الشاشة وليكن مثلا :

Delete For Age > 25

ويعنى ذلك مسح سجلات الطلبة الذين تزيد أعمارهم عن ٢٥ سنة.

٧- يتم الضغط على مفتاح الإدخال لإدخال هذا الشرط ثم يتم تحريك العمود الضوئى (Highlight) الخاص بقائمة البحث حتى يصل إلى(Execute the Command) ثم الضغط على مفتاح الإدخال.

٨- يلاحظ ظهور رسالة أسفل الشاشة توضح عدد السجلات التى يتم مسحها. وهى فى الواقع لايتم مسحها نهائيا ولكن يتم وضع علامات عليها حتى يتم مسحها نهائيا بعد ذلك باستخدام الأمر (PACK).

٩- للتأكد من وجود علامات أمام السجلات المطلوب مسحها ، يتم استخدام الاختيار (Browse) لعرض سجلات الملف. ويلاحظ وجود علامات أمام السجلات التى تم تجهيزها للمسح.

١٥ - ٦ الإستعادة (Recall)

ويستخدم هذا الاختيار لاستعادة بعض السجلات التى سبق وضع علامات بها حتى لايتم مسحها بواسطة الاختيار (Pack). ويتم تنفيذ ذلك باستخدام نفس الخطوات التى سبق استخدامها مع الاختيار (Delete) لتحديد السجلات التى تحقق شرطا أو شروطا معينة. كما يمكن تحديد سجل معين برقمه والتأكد من إختفاء العلامة أمامه عن طريق استخدام الاختيار (Browse) كما سبق الإيضاح.

١٥ - ٧ المسح النهائى (Pack)

ويستخدم هذا الاختيار لمسح السجلات التى سبق وضع علامات أمامها تمهيدا لمسحها. حيث أن السجلات التى سبق تحديدها ووضع علامات أمامها تظل موجودة ويمكن عرضها على الشاشة وتعديلها. وعند استخدام الأمر (Pack) يتم مسحها نهائيا. ولذلك يلزم قبل استخدام الامر (PACK) التأكد من أن السجلات التى تم تمييزها بعلامات لمسحها هى السجلات المطلوب مسحها فعلا.

ولتنفيذ هذه العملية يتم تحريك العمود الضوئى (Hilghlight) حتى يصل إلى الاختيار (Pack) والضغط على مفتاح الإدخال. يلاحظ فى هذه الحالة ظهور رسالة أسفل الشاشة توضح عدد السجلات التى يتم نسخها. حيث أن الأمر (Pack) يؤدى إلى نسخ جميع سجلات ملف قاعدة البيانات ماعدا السجلات التى تم وضع علامات عليها لمسحها. وفى هذه الحالة يجب إعادة إنشاء الفهرس إذا كان قد سبق إنشاء فهرس للملف.

الفصل السادس عشر

تنظـــيم الملــف

(File Organization)



المقصود بتنظيم الملف هو ترتيب السجلات داخل هذا الملف بطريقة تسهل البحث خلاله والوصول إلى المعلومات المطلوبة بسرعة وسهولة.

وعند إنشاء ملف قاعدة البيانات لأول مرة فإن السجلات يتم تخزينها بنفس ترتيب إدخالها أى أن ترتيبها لايعتمد على حقل معين. وعندما يراد البحث عن سجل معين فإن البحث دائما يعتمد على محتويات حقل معين مثل البحث عن طالب إسمه فتحى مثلا ، أو الطالب الذى يسكن فى عنوان معين وهكذا. وفى هذه الحالة يقوم البرنامج بالبحث خلال جميع السجلات ومقارنة بيانات حقل الإسم مثلا بالاسم المطلوب حتى يصل إلى السجل الخاص بهذا الطالب. وهذه العملية تأخذ وقتا طويلا نتيجة لأن السجلات غير مرتبة بناء على حقل الإسم بالترتيب الهجائى مثلا. أما اذا كانت مرتبة هجائيا حسب الإسم فإن البرنامج يبحث فى ترتيب الحروف حتى يصل إلى حرف (F) ثم يبحث عن الإسم المطلوب حسب الترتيب الهجائى للحروف التالية للحرف (F). وفى هذه الحالة تصبح عملية البحث سهلة وسريعة. وبالمثل يمكن ترتيب السجلات حسب أى حقل آخر عندما يراد البحث عن سجل معين عن طريق بيانات هذا الحقل.

وبرنامج (+ DBase III) يتيح طريقتين لتنظيم الملف أحدهما تسمى الفرز (Sorting) والأخرى تسمى الفهرسة (Indexing).

وفى الأجزاء التالية من هذا الفصل يتم إلقاء الضوء على هاتين الطريقتين وخصائص كل منهما.

١٦ - ١ الفـــرز (Sorting)

الفرز هو طريقة لترتيب السجلات داخل الملف حسب بيانات حقل معين وذلك بتغيير المواقع الفعلية للسجلات فى الملف. والطريقة الوحيدة لتنفيذ ذلك هى نسخ الملف بأكمله مع تغيير مواقع السجلات به ، أى أن الفرز يتطلب دائما إنشاء ملف جديد. وعند إضافة سجلات جديدة إلى الملف الذى تم فرزه فإن هذه السجلات توضع بعد آخر سجل فى الملف وبالتالى لاتوضع فى ترتيبها حسب الحقل الذى تم الترتيب بناء عليه. وفى هذه الحالة يلزم إعادة فرز الملف مرة ثانية. ومع كل فرز جديد يتم إنشاء ملف جديد بالإضافة إلى الملف الأصلى مما يسبب تحميلا كبيرا (Overload) على ذاكرة الحاسب. ورغم عيوب الفرز التى سبق إيضاحها إلا انه أحيانا يكون مطلوبا ، وذلك عندما يراد مثلا الحصول على ملف مرتب حسب حقل معين بترتيب تنازلى (Descending) بدلا من الترتيب التصاعدى

(Ascending). حيث أن الفرز يتيح الترتيب التنازلى ولكن عن طريق الأوامر (Commands) وليس عن طريق برنامج المساعد (Assistant) كما سيتم الإيضاح فيما بعد.

ويجب ملاحظة أن الفرز يغير أرقام السجلات نتيجة نقل السجلات فى أماكن أخرى. فمثلا اذا كانت هناك مجموعة من السجلات التى تحتوى على بيانات طلبة ويراد فرزهم أبجديا بالترتيب التصاعدى (Ascending) باستخدام حقل الإسم فإن الشكل التالى يوضح الأسماء قبل الفرز وبعده كالآتى :

الجدول قبل الفرز			الجدول بعد الفرز		
رقم السجل	الاسم	العمر	رقم السجل	الاسم	العمر
١	محمود	١٨	١	أحمد	١٩
٢	عمر	١٧	٢	بهجت	١٦
٣	أحمد	١٩	٣	سالم	١٠
٤	سالم	٢٠	٤	عمر	١٧
٥	بهجت	١٦	٥	محمود	١٨

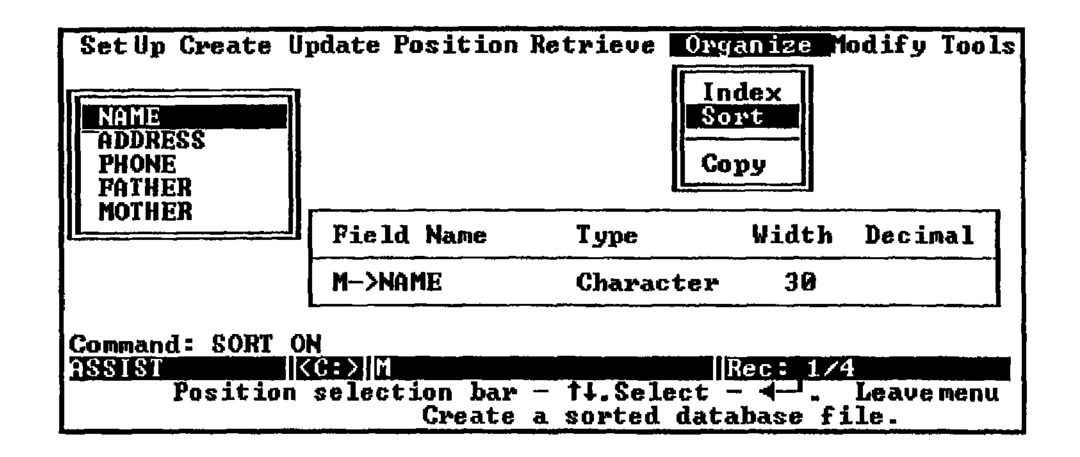
يلاحظ من الجدول أن محمود كان رقمه (١) فأصبح (٥) وعمر كان رقمه (٢) فأصبح (٤) وهكذا. أى أن أرقام السجلات تغيرت بعد الفرز.

ولتنفيذ عملية الفرز من خلال برنامج المساعد (Assistant) يتم اتباع الخطوات التالية :

١- يتم فتح قائمة التنظيم (Organize) والتى تحتوى على ثلاثة إختيارات منها الإختيار (Sort).
٢- يتم تحريك العمود الضوئى حتى يصل إلى الإختيار (Sort) والضغط على مفتاح الإدخال.
٣- يلاحظ ظهور قائمة فرعية تحتوى على أسماء الحقول الموجودة فى الملف والتى يتم منها اختيار الحقول التى يتم الترتيب بناء عليها.
٤- بعد اختيار الحقول يتم الضغط على مفتاح السهم يمين للخروج من قائمة الحقول. ثم

يتم تحديد إسم الملف الذى يتم فرزه وفى هذه الحالة يمكن كتابة إسم جديد للملف لإنشاء ملف جديد مع الإحتفاظ بالملف الأصلى.

٥- يلاحظ ظهور رسالة أسفل الشاشة توضح عدد السجلات التى تم فرزها ، أنظر الشكل (١٦ - ١).



الشكل (١٦ - ١) تنظيم الملف

١٦ - ٢ الفهرسة (Indexing)

الفهرسة هى طريقة تستخدم لترتيب السجلات مثل الفرز ولكنها تختلف عن الفرز فى أنها لاتغير المواقع الفعلية للسجلات. وهى تعتمد على إنشاء فهرس مكون من حقلين فقط أحدهما يحتوى على أرقام السجلات (Record Numbers) والآخر يحتوى على البيانات المطلوب الترتيب بناء عليها مثل الإسم أو الرقم أو ... الخ. وهذا الحقل الآخر يكون مرتبا ترتيبا تصاعديا (Ascending).

فعندما يراد البحث عن سجل معين بناء على حقل الإسم مثلا يتم البحث فى فهرس الإسم عن هذا الإسم وبالتالى يتم تحديد رقم السجل الخاص به. وعن طريق رقم السجل يمكن الوصول مباشرة إلى السجل المطلوب.

ولتوضيح ذلك نفرض أن نفس السجلات المستخدمة فى المثال السابق يراد عمل فهرس لها حسب الإسم فيصبح الفهرس كالآتى :

الفهرس		الملف		
الاسم	رقم السجل	العمر	الإسم	رقم السجل
أحمد	٣	١٨	محمود	١
بهجت	٥	١٧	عمر	٢
سالم	٤	١٩	أحمد	٣
عمر	٢	٢٠	سالم	٤
محمود	١	١٦	بهجت	٥

فعندما يراد الوصول إلى السجل الخاص بأحمد مثلا يتم تحديد رقم السجل الخاص به من الفهرس وعن طريق هذا الرقم يمكن الوصول إلى السجل الخاص به مباشرة. و يلاحظ هنا أن أرقام السجلات تظل كما هى دون تغيير.

وعملية الفهرسة تشبه استخدام فهرس الكتاب للوصول إلى موضوع معين. حيث يتم أولا البحث فى الفهرس عن هذا الموضوع. وعند الوصول إليه يتم تحديد رقم الصفحة التى تحتوى على هذا الموضوع. وعن طريق رقم الصفحة يمكن الوصول إلى الموضوع مباشرة. ورقم الصفحة فى هذه الحالة يقابل رقم السجل في ملف قاعدة البيانات.

ولتنفيذ عملية الفهرسة يتم اتباع الخطوات التالية :

١ - يتم فتح قائمة التنظيم (Organize).
٢ - يتم تحريك العمود الضوئى (Highlight) حتى يصل إلى الإختيار (Index).
٣ - يلاحظ ظهور الرسالة التالية :

Enter an Index Key Expression

والمقصود هنا الحقل المطلوب استخدامه فى الفهرس. ويمكن كتابة إسم هذا الحقل الفهرسى أو اختياره من قائمة الحقول التى تظهر عند الضغط على مفتاح (F10) أو مفتاحى (Ctrl-Home).

٤ - يمكن استخدام عدة حقول فى الفهرس الواحد وذلك بكتابة إسم الحقل الأول ثم علامة

الجمع (+) ثم إسم الحقل الثانى و ... وهكذا. وهذا يعنى إستخدام الحقل الأول كمفتاح رئيسى والحقل الثانى كمفتاح ثانوى و ... وهكذا. أى أن السجلات يتم ترتيبها حسب الإسم مثلا. وعند تطابق عدة أسماء يتم ترتيبهم حسب العمر مثلا.

٥ - يلاحظ ظهور الرسالة التالية على الشاشة :

Enter the Name of the File

فيتم كتابة الإسم ويفضل فى هذه الحالة إختيار إسم يوضح نوع الفهرس المستخدم. فمثلا عند إنشاء فهرس للأسماء يمكن تسميته (Name) مع ملاحظة أن البرنامج يضيف إليه الإمتداد (Extension) الذى يكون فى جميع الأحوال (NDX.).

٦- يمكن إنشاء عدة ملفات فهرس (Index Files) للحصول على ترتيب مختلف للسجلات حسب الحاجة. ويمكن استخدام أى نوع من الحقول فى الفهرس ماعدا الحقول المنطقية (Logical) وحقول الملاحظات (Memo). ويمكن جمع عدة حقول فى الحقل الفهرسى (Key Field) ولكن يشترط فى هذه الحالة أن تكون جميعها من نفس النوع. فمثلا إذا كان أحد الحقول حرفيا فيجب أن تكون باقى الحقول المجموعة عليه حرفية أيضا. وعندما يراد جمع حقل حرفى مع حقل تاريخى (Date) مثلا يجب أولا تحويل حقل التاريخ إلى حقل حرفى باستخدام دالة خاصة (Function) تقوم بعملية التحويل.

فمثلا يمكن استخدام الحقل التالى كحقل فهرسى

Name + DTOC (Birth_d)

حيث تستخدم الدالة (DTOC) لتحويل التاريخ إلى حروف. كما سيتم الإيضاح فى الجزء الخاص بالدوال (Functions).

١٦ - ٢ - ١ إستخدام ملف الفهرس

كما سبق الإيضاح فإنه يمكن إنشاء أى عدد من ملفات الفهرس المرتبطة بملف قاعدة بيانات واحد. ولكن لايمكن فتح أكثر من سبع ملفات فهرس فى نفس الوقت مع ملف قاعدة البيانات. ويعتبر أول ملف يتم فتحه هو الملف الرئيسى (Master) وباقى الملفات ثانوية. وتتم عملية فتح ملفات الفهرس حسب الخطوات التالية :

١- يتم فتح قائمة التجهيز (Set Up) ويكون المؤشر واقفا عند أول اختيار وهو (Database File) فيتم الضغط على مفتاح الإدخال.
٢- يظهر سؤال عن وحدة الأقراص المطلوب استخدامها فيتم إدخالها.
٣- تظهر ملفات قواعد البيانات الموجودة على وحدة الأقراص المستخدمة فيتم اختيار الملف المطلوب فتحه.
٤- يظهر على الشاشة السؤال التالى :

Is the file indexed ? (Y/N)

٥- يتم كتابة (Y).
٦- تظهر قائمة بملفات الفهرس التى سبق إنشاؤها إذا كان هناك أكثر من فهرس لملف قاعدة البيانات المفتوح.
٧- يتم اختيار الفهرس المطلوب فتحه ويلاحظ فى هذه الحالة ظهور كلمة(Master) أمام إسم الملف المفتوح. ويعنى ذلك استخدام هذا الملف كفهرس رئيسى فى ترتيب السجلات. وإذا تم استخدام أكثر من فهرس يكون الأول رئيسيا والملفات الباقية ثانوية حسب ترتيبها.

الفصل السابع عشر

البحث

(Query)



عندما يراد استرجاع أى معلومات من قاعدة البيانات فإن ذلك يتطلب البحث عن السجل الذى يحتوى على هذه المعلومات. ويتم هذا البحث إما بناء على قيمة معينة فى الحقول تحقق شروطا معينة أو عن طريق رقم السجل الذى يمكن عن طريقه الوصول إلى سجل محدد.

١٧ - ١ استخدام مؤشر السجلات (Record Pointer)

مؤشر السجلات هو مؤشر منطقى (Logical) يشير إلى سجل معين ولكنه لايظهر على الشاشة. وعند إجراء عرض أو تعديل للبيانات تظهر البيانات الخاصة بالسجل الذى يقف عنده المؤشر. وعن طريق توجيه هذا المؤشر يمكن الوصول إلى سجل معين.

ولتوجيه المؤشر إلى سجل معين تتبع الخطوات التالية :

١ - يتم فتح قائمة المكان (Position).
٢ - يتم تحريك العمود الضوئى(Highlight) للوصول إلى الإختيار(Goto Record).
٣ - يلاحظ أن عمود الحالة (Status Bar) يشير إلى السجل رقم (١) وهو الوضع المبدئى (Default) لمؤشر السجلات (Record Pointer).
٤ - يلاحظ ظهور قائمة فرعية تتضمن ثلاثة إختيارات وهى :

Record , Bottom , Top

٥ - عند اختيار (Top) فإن مؤشر السجلات (Record Pointer) يظل عند أول سجل فى ملف قاعدة البيانات لأنه يمثل قمة الملف.
٦ - عند اختيار (Bottom) يذهب المؤشر إلى آخر سجل فى ملف قاعدة البيانات.
٧ - عند اختيار (Record) يظهر سؤال عن رقم السجل المراد الذهاب إليه فيتم كتابة الرقم والضغط على مفتاح الإدخال.
٨ - يتم الخروج من قائمة المكان (Position).
٩ - يمكن الدخول إلى قائمة التحديث (Update) واختيار (Edit) ويلاحظ ظهور بيانات هذا السجل الذى تم توجيه المؤشر إليه.
١٠- يمكن تحريك المؤشر عدة خطوات وذلك بالرجوع إلى قائمة المكان (Position) واختيار (Skip) فيلاحظ ظهور الرسالة التالية :

Enter a Numeric Value :

١١- يتم كتابة عدد السجلات المراد تخطيها أمام هذه الرسالة وليكن (4) مثلا والضغط على مفتاح الإدخال. فى هذه الحالة ينتقل المؤشر بعد السجل الذى كان يقف عنده بأربعة سجلات ويقف عند السجل الجديد. فإذا كان المؤشر واقفا فى البداية عند السجل رقم (10) فانه ينتقل إلى السجل رقم (14).

ملاحظة

للقفز (Skip) عددا من السجلات فى الإتجاه العكسى (أى فى إتجاه قمة الملف) يتم استخدام إشارة (-) قبل الرقم المطلوب إدخاله. فمثلا عندما يكون المؤشر أصلا عند السجل رقم (10) وتم كتابة (4-) فإن المؤشر ينتقل إلى السجل رقم (6).

١٧ - ٢ توجيه المؤشر إلى سجل يحقق شروطا معينة

تم فى الجزء السابق شرح طريقة توجيه المؤشر إلى سجل معين عن طريق رقم السجل (Record Number). ولكن فى معظم الأحيان يكون رقم السجل غير معلوم. وذلك لأن البرنامج يقوم بتحديد رقم السجل بناء على الترتيب الفعلى لإدخال السجلات. وعند حدوث أى تعديل فى هذا الترتيب عن طريق الفرز مثلا (Sorting) فإن رقم السجل السابق لايصبح مرتبطا بنفس السجل ولكنه يشير إلى سجل آخر. أى أن المستخدم لايستطيع متابعة رقم السجل ومعرفة الرقم المقابل لكل سجل. وفى هذه الحالة يلزم الإعتماد على البيانات المخزنة فى الحقول ووضع شروط معينة للقيم الموجودة فى هذه الحقول للوصول إلى السجل أو السجلات التى تحقق هذه الشروط. وتوجد عدة طرق لتنفيذ ذلك منها إستخدام الأمر (Locate) الموجود فى قائمة المكان (Position).

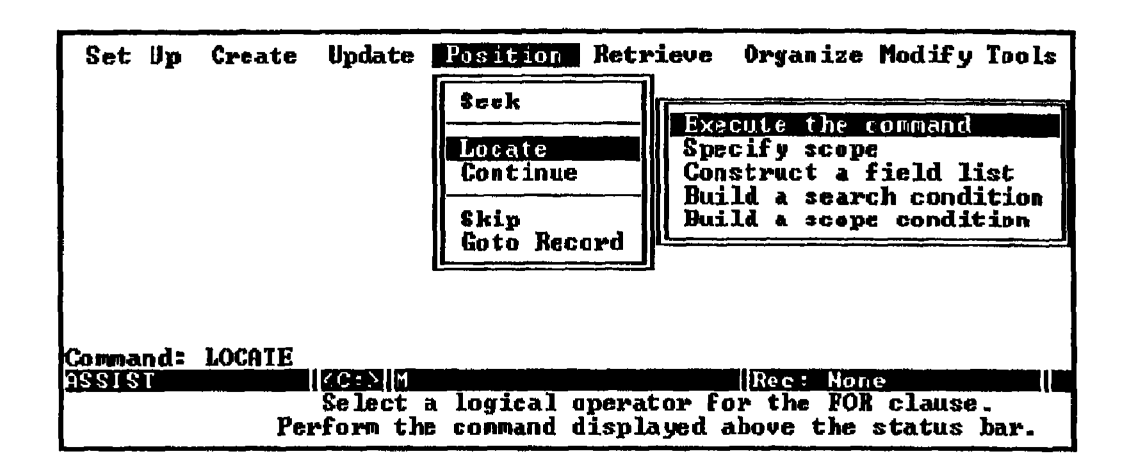
١٧ - ٣ استخدام الأمر (Locate) فى الوصول إلى سجل محدد

عندما يراد الوصول إلى سجل محدد باستخدام الأمر (Locate) فإن ذلك يتم عن طريق تحديد قيمة لحقل أو عدة حقول يراد البحث عنها. ويقوم البرنامج بمقارنة هذه القيمة بجميع القيم الخاصة بهذا الحقل أو هذه الحقول لجميع السجلات. وعندما يجد السجلات المطابقة فانه يضع المؤشر عند أول سجل مطابق.

ويمكن استخدام الأمر (continue) بعد ذلك للإنتقال إلى السجل الذى يليه وهكذا. ولتنفيذ ذلك تتبع الخطوات التالية :

١- يتم فتح قائمة المكان (Position) واختيار (Locate).

٢- يلاحظ ظهور قائمة فرعية تحتوى على عدة اختيارات مع وقوف العمود الضوئى (Highlight) على أول اختيار فى القائمة. انظر شكل (١٧ - ١).



شكل (١٧ - ١)

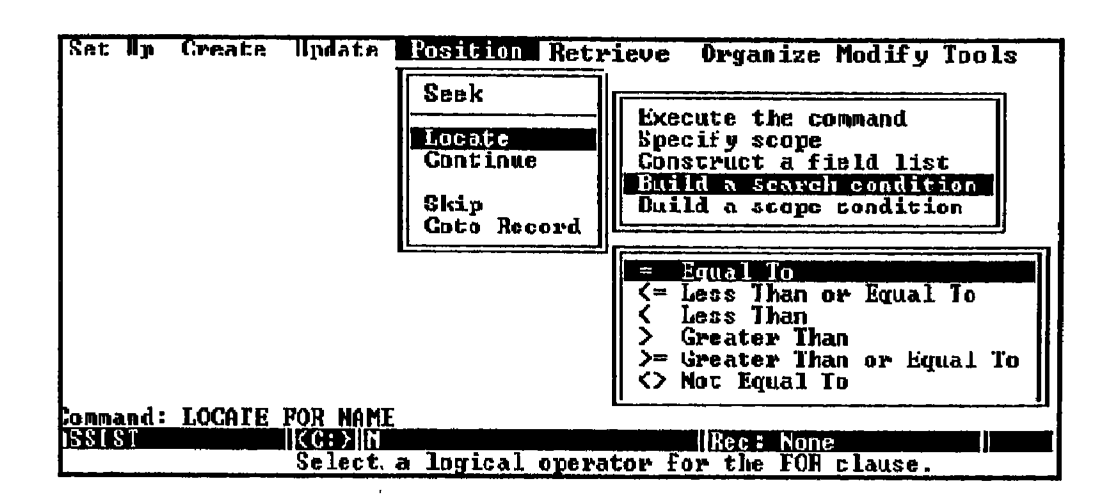
٣- يتم تحريك العمود الضوئى (Highlight) باستخدام مفتاح الإتجاه (|) حتى يصل إلى الإختيار (Build a Search Condition) والضغط على مفتاح الإدخال.

٤- يلاحظ ظهور قائمة بأسماء الحقول الموجودة فى ملف قاعدة البيانات(DBase File) فيتم اختيار الحقل المطلوب استخدامه فى تحديد السجل أو السجلات المطلوبة وليكن حقل الإسم (Name).

٥- يلاحظ ظهور قائمة بمعاملات المقارنة المطلوب استخدامها فى تكوين الشرط المطلوب تحقيقه. انظر الشكل (١٧ - ٢)

٦- يتم اختيار معامل المقارنة المطلوب وليكن (= Equal To).

٧- يظهر سؤال عن القيمة المطلوب مقارنتها فيتم إدخالها مع ملاحظة أن القيمة يتم إدخالها بدون علامات تنصيص (Quotation) حتى لو كانت حرفية (String). وليكن الإسم المطلوب البحث عنه مثلا هو (Mohamed). فيتم كتابة الإسم والضغط على مفتاح الإدخال.



شكل (١٧ - ٢)

٨- يتم تحريك العمود الضوئى (Highlight) حتى يصل إلى الإختيار (No More Conditions) والضغط على مفتاح الإدخال.

٩- يلاحظ ظهور الأمر التالى على خط الأوامر :

Locate For Name = Mohamed

وهو يمثل الأمر المناظر للإختيارات التى تم تحديدها من القوائم.

١٠- يتم اختيار (Execute the Command).

١١- يلاحظ ظهور رقم أول سجل يحقق هذا الشرط على عمود الحالة (Status Bar). وهذا يعنى أن المؤشر يقف الآن عند هذا السجل. فإذا أريد تعديل بيانات هذا السجل تستخدم القوائم فى ذلك كما سبق الإيضاح.

١٢- إذا أريد الوصول إلى سجل آخر يحقق الشرط يتم تحريك العمود الضوئى (Highlight) إلى الاختيار (Continue) والضغط على مفتاح الإدخال فيلاحظ تغير رقم السجل المكتوب فى عمود الحالة (Status Bar). وهكذا يمكن الوصول إلى جميع السجلات التى تحقق الشرط.

١٣- يمكن استخدام عدة شروط فى البحث عن السجل بالربط بين الشروط بواسطة المعاملات المنطقية (Logical Operators) مثل (OR , AND) التى تظهر فى القائمة الفرعية.

ملاحظة

يمكن استخدام معاملات المقارنة مثل أكبر من (>) ، اصغر من (<) مع المدخلات الحرفية (String). وفى هذه الحالة يتم مقارنة الحرف الأول فى القيمتين حسب ترتيبه فى الترتيب الهجائى للحروف.

١٧ - ٤ إسترجاع السجلات (Retrieving)

يمكن عن طريق قائمة الإسترجاع (Retrieve) عرض بيانات عدد من السجلات التى تحقق شرطا أو شروطا معينة ويستخدم لذلك الأمر (List) ، والأمر (Display).

ولتنفيذ ذلك يتم اتباع الخطوات التالية :

١- يتم فتح قائمة الإسترجاع (Retrieve) واختيار الأمر (List).
٢- يتم اختيار الأمر (Construct a Field List).
٣- يلاحظ ظهور قائمة بحقول الملف فيتم اختيار الحقول المراد عرض بياناتها.
٤- يتم الضغط على مفتاح السهم يمين (- ->) للخروج من قائمة الحقول.
٥- يتم اختيار (Build a Search Condition) ويتم اختيار الحقل المطلوب استخدامه فى شرط البحث.
٦- يلاحظ ظهور قائمة معاملات المقارنة.
٧- يتم اختيار المعامل المطلوب والضغط على مفتاح الإدخال.
٨- يلاحظ ظهور سؤال عن القيمة المطلوب مقارنتها فيتم إدخالها والضغط على مفتاح الإدخال.
٩- إذا أريد إدخال شرط آخر يتم اختيار المعامل المنطقى المطلوب استخدامه إذا كان (AND) أو (OR).
١٠- يتم اختيار الأمر (Execute the Command).
١١- يلاحظ ظهور السؤال التالى :

Direct The Output To The Printer ? (Y/N)

فإذا أريد عرض السجلات على الشاشة فقط يتم كتابة (N) أما إذا أريد طباعة هذه السجلات فيتم كتابة (Y).

ملاحظة

يمكن استخدام الإختيار (Display) بدلا من الإختيار (List) لتحقيق نفس النتيجة. والفرق بينهما أن (List) فى الوضع المبدئى له يؤدى إلى عرض جميع سجلات ملف قاعدة البيانات. أما الإختيار (Display) فإن الوضع المبدئى له يؤدى إلى عرض سجل واحد فقط وهو السجل الذى يقف عنده المؤشر. ومع ذلك فإن إدخال شروط معينة فى الحالتين يؤدى إلى الوصول إلى نفس النتيجة.

الفصل الثامن عشر

ملفــات البحــث

(Query Files)



عندما يريد المستخدم استرجاع مجموعة محددة من السجلات التى تحقق نفس الشروط (فمثلا عندما يراد دائما البحث خلال سجلات الموظفين الذين التحقوا بالعمل إبتداء من سنة ١٩٨٠ وتزيد أعمارهم عن ٣٠ سنة) فبدلا من تكرار إدخال هذه الشروط عند كل عملية بحث عن أى موظف فمن الأفضل إنشاء ملف بحث (Query File) وإدخال كل الشروط المطلوبة فيه. ويستخدم هذا الملف كمرشح (Filter) يتم من خلاله تصفية قاعدة البيانات وعدم السماح بالمرور من هذا المرشح إلا للسجلات التى تحقق الشروط الموجودة به. ويمكن تخزين هذا الملف واستخدامه وقت الحاجة. كما يمكن إنشاء عدة ملفات بحث واستخدام أى ملف منها مع ملف قاعدة البيانات. وهذه الطريقة تتيح للمستخدم مرونة كاملة فى التعامل مع السجلات.

١٨ - ١ إنشاء ملف البحث

الهدف من ملف البحث كما سبق الإيضاح هو تصفية عدد السجلات التى يتم عرضها والتعامل معها وذلك عن طريق إستبعاد السجلات التى لاتحقق شروطا معينة. ويتم إدخال هذه الشروط عن طريق نموذج خاص كما يتضح من الشكل (١٨ - ١) وعن طريق اتباع الخطوات التالية :

Set Filter   Nest   Display   Exit

Field Name
Operator
Constant/Expression
Connect

Line Number   1

Line	Field	Operator	Constant/Expression	Connect
1				
2				
3				
4				
5				
6				
7				

Position selection bar - ↑↓. Select - ←┘. Leave menu - ←→.
Select a field name for the filter condition.

شكل (١٨ - ١)

١- يتم فتح قائمة التجهيز (Set Up) واختيار ملف قاعدة البيانات (DBase File) المطلوب فتحه.
٢- يظهر سؤال عما إذا كان الملف تم فهرسته (Indexed) أم لا. وفى حالة فهرسة الملف يتم كتابة إسم ملف الفهرس (Index File).
٣- يتم الخروج من قائمة التجهيز (Set Up) عن طريق مفتاح السهم يمين (<--).
٤- يتم فتح قائمة البحث (Query) ويتم تحديد وحدة الأقراص الموجود بها القرص المطلوب تخزين هذا الملف به.

يلاحظ ظهور عمود الإختيارات الخاصة بالبحث. وهذا العمود يحتوى على إختيارات يتم عن طريقها تحديد شروط البحث. كما يلاحظ ظهور جدول (Table) يتم فيه وضع الحقول والقيم المطلوب مقارنتها ومعاملات المقارنة. وهذا الجدول يتيح إدخال عدة شروط كما يتيح الربط بين هذه الشروط باستخدام المعاملات المنطقية (AND) و (OR) لتكوين شروط مركبة.

ولتمثيل شرط مركب فى هذا الجدول يتم أولا كتابة هذا الشرط خارج الجهاز. فمثلا إذا أريد إدخال شرط يحدد الموظفين الذين التحقوا بالعمل بعد ١ - ١ - ١٩٩٠ وتزيد أعمارهم عن ٣٠ سنة أو تزيد مرتباتهم عن ٣٠٠ جنيه بشرط أن يكونوا متزوجين يتم كتابة هذا الشرط خارج الجهاز كالآتى :

$$\text{DAT_ENT} > 1/1/1990 \text{ AND } \begin{bmatrix} \text{AGE} > 30 \\ \text{OR} \\ \text{SALARY} > 300 \end{bmatrix} \text{ AND MARRIED}$$

وذلك بفرض أن (DAT_ENT) يمثل إسم الحقل الخاص بتاريخ إلتحاق الموظف و (AGE) يمثل إسم الحقل الخاص بعمر الموظف و (SALARY) يمثل مرتب الموظف و (MARRIED) هو إسم الحقل المنطقى (Logical) الذى يوضح إذا كان الموظف متزوجا أو غير متزوج.

ولكتابة هذا الشرط المركب بالصورة التى يميزها برنامج (+ DBase III) يصبح كالآتى :

DAT_ENT > 1/1/1990 AND ((AGE > 30) OR (SALARY > 300)) AND MARRIED.

ويلاحظ هنا استخدام الأقواس (Parentheses) لتحديد ترتيب تنفيذ البرنامج لعمليات المقارنة.

ولكتابة هذا الشرط بالجدول تتبع الخطوات الآتية : أنظر الشكل (١٨ - ٢)

١- عند ظهور عمود الإختيارات (Menu Bar) الخاص بقائمة البحث (Query) يلاحظ وقوف المؤشر عند أول اختيار فى القائمة وهو (Set Filter) وبالتالى فتح القائمة الخاصة به. ويلاحظ كذلك وقوف العمود الضوئى (Highlight) عند أول اختيار فى القائمة وهو (Field Name) فيتم الضغط على مفتاح الإدخال.

```
Set Filter            Nest              Display              Exit
  Field Name  ►                                       NAME
  Operator                                            ADDRESS
  Constant/Expression                                 PHONE
  Connect                                             FATHER
                                                      MOTHER
  Line Number        1

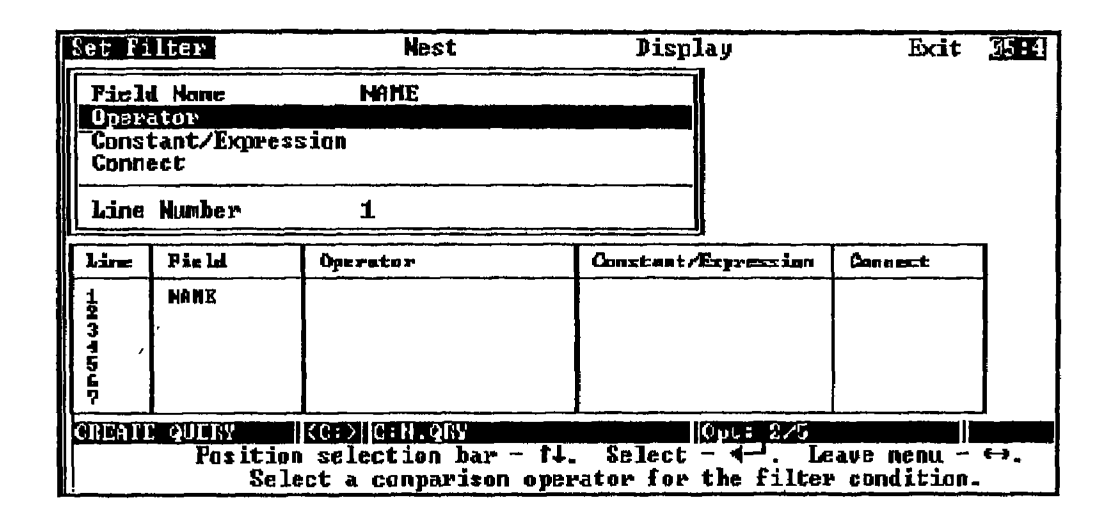
  Field Name         Type        Width  Decimal  ant/Expression   Connect
  M->NAME            Character    30
  3
  4
  5
  6
  7
CREATE QUERY  |<C:>|C:M.QBE|          |Opt: 1/5|        |Num
   Position selection bar - ↑↓.  Select - ◄┘.  Leave menu - ←→.
          Select a field name for the filter condition.
```

شكل (١٨ - ٢)

٢- يلاحظ ظهور قائمة بأسماء الحقول الموجودة بالملف فيتم اختيار الحقل الأول فى الشرط وهو (Name) فيلاحظ كتابة إسم هذا الحقل فى عمود الحقل (Field) فى الجدول.

٣- يتم تحريك العمود الضوئى إلى الإختيار (Operator) فيلاحظ ظهور قائمة بمعاملات المقارنة (Relational Operators) فيتم اختيار المعامل المطلوب وهو

(> More Than) فيلاحظ ظهور هذا المعامل فى عمود المعامل (Operator) فى الجدول. انظر الشكل (١٨ - ٣)



شكل (١٨ - ٣)

٤- يتم تحريك المؤشر الضوئى إلى الإختيار (Constant / Expression) وكتابة التاريخ كالآتى :

CTOD ("01/01/1990")

ويلاحظ هنا استخدام الدالة (CTOD) لتحويل التاريخ إلى قيمة يميزها البرنامج. وهذا سوف يتم إيضاحه فيما بعد.

٥- يتم تحريك المؤشر الضوئى إلى الإختيار (Connect) وكتابة المعامل المنطقى المطلوب للربط بين هذا الشرط والشرط التالى له. وفى هذه الحالة يتم كتابة المعامل المنطقى (AND).

٦- يتم إضافة الشروط الأخرى بنفس الطريقة مع الربط بينها بالمعامل المنطقى المطلوب.

٧- يتم الضغط على مفتاح (<--) للخروج من قائمة (Set Filter) ويلاحظ إنتقال مؤشر عمود الإختيارات (Menu Bar) إلى الإختيار التالى وهو (Nest) والذى يسمح باستخدام شروط مركبة عن طريق الأقواس كما سيتم الإيضاح فى الجزء التالى.

١٨ - ٢ تداخل الشـروط (Nesting)

عندما يبحث برنامج (+DBase III) عن سجل محدد بناء على شروط معينة متداخلة فإنه يتبع قواعد الأسبقية (Precedence Rules) المعروفة فى معظم لغات الحاسب لتحديد ترتيب تنفيذ هذه الشروط.

وعن طريق استخدام الأقواس يمكن التحكم فى أولويات تنفيذ الشروط فى العلاقة. لذلك يتم وضع الأقواس حول كل شرط يراد تنفيذه قبل الشروط الأخرى.

ففى المثال السابق تم وضع العلاقة بالصورة التالية :

DAT_ENT > 1/1/1990 AND ((AGE> 30) OR (SALARY> 300))
AND MARRIED

ولإضافة الأقواس فى العلاقة تتبع الخطوات التالية :

١ - يتم فتح قائمة (Nest) فيلاحظ وقوف العمود الضوئى (Highlight) الخاص بها عند الإختيار (Add) فيتم اختيار (Start) وكتابة رقم السطر الذى يتم وضع أول قوس عنده ثم اختيار (End) وكتابة رقم السطر الذى يتم كتابة نهاية القوس عنده ويلاحظ ظهور الأقواس على الجدول.
٢ - يمكن اتباع نفس الخطوات لكتابة أى أقواس أخرى خارجية.

١٨ - ٣ عرض وتخزين ملف البحث (Query File)

بعد إدخال شروط البحث فى الجدول كما سبق الإيضاح يلزم أولا التأكد أن هذه الشروط سوف تؤدى إلى إختيار السجلات المطلوبة. ويتم ذلك عن طريق الإختيار (Display) من قائمة الإختيارات (Menu Bar). ولتنفيذ ذلك يتم اتباع الخطوات التالية :

١ - يتم استخدام مفتاح السهم يمين (<--) للإنتقال إلى الإختيار (Display) والضغط على مفتاح الإدخال.
٢ - يلاحظ ظهور بيانات أول سجل يحقق شروط البحث.

انظر الشكل (١٨ - ٤)

```
Set Filter                 Nest            Display              Exit
AME          mohamed hasan fathy
DDRESS       12-ain shams
HONE         56526756
ATHER        hasan fathy
OTHER        samua tawfik
```

Line	Field	Operator	Constant/Expression	Connect
1	NAME	Begins with	"m"	.OR.
2	ADDRESS	Matches	"ain shams"	
3				
4				
5				
6				
7				

```
CREATE QUERY    |<C:>|C:N.QRY                |Rec: 1/2
Next/Previous record - PgDn/PgUp.  Toggle query form - F1.  Leave option -
         Display records in the database that meet the query condition.
```

شكل (١٨ - ٤)

ملاحـظة

عند وجود أى خطأ فى جدول البحث تظهر الرسالة التالية :

Invalid Filter

ولايتم ظهور أى سجلات وفى هذه الحالة يتم الرجوع إلى قائمة تجهيز المرشح (Set Filter) وتصحيح الأخطاء الموجودة قبل تخزين ملف البحث.

٣- يمكن الضغط على مفتاح (PgDn) لعرض السجل التالى والذى يليه وهكذا.

٤- بعد عرض عدة سجلات والتأكد أنها تحقق الشروط يتم الضغط على مفتاح السهم يمين (<---) للخروج من قائمة العرض (Display) وفتح القائمة التالية لها وهى قائمة الخروج (Exit).

٥- من قائمة الخروج يتم اختيار الأمر (Save) فيتم تخزين ملف البحث بنفس الإسم الذى سبق تحديده. انظر الشكل (١٨ - ٥)

```
Set Filter              Nest              Display            Exit  05:43:23 pm
                                                              Save
                                                              Abandon

| Line | Field   | Operator    | Constant/Expression | Connect |
| 1    | NAME    | Begins with | "n"                 | .OR.    |
| 2    | ADDRESS | Matches     | "ain shams"         |         |
| 3    |         |             |                     |         |
| 4    |         |             |                     |         |
| 5    |         |             |                     |         |
| 6    |         |             |                     |         |
| 7    |         |             |                     |         |

CREATE QUERY  |<C:>|C:N.QRY          |Opt: 1/2
   Position selection bar - ↑↓.  Select - ◄┘.  Leave menu - ↔.
                      Exit and save changes.
```

شكل (١٨ - ٥)

١٨ - ٤ إستخدام ملف البحث

عندما يراد استخدام ملف البحث يتم فتحه من خلال قائمة التجهيز (Set Up) حيث يتم اختيار إسم قاعدة البيانات أولا ثم اختيار ملف البحث الذى سبق إنشاؤه. ويمكن إنشاء عدة ملفات بحث وتخزينها ثم اختيار ملف البحث المطلوب فى كل مرة يتم فيها فتح ملف قاعدة البيانات (DBase File).

١٨ - ٥ المعاملات الحرفية (Character Operators)

كما سبق الإيضاح فإنه عند كتابة معاملات المقارنة فإن ذلك يتم لكل حقل يتم اختياره من ملف قاعدة البيانات (DBase File). ولذلك فإن ما يظهر من هذه المعاملات هو المعاملات التى تخص نوع الحقل المستخدم سواء كان عدديا أو حرفيا أو تاريخيا أو ... الخ. ويقوم المستخدم باختيار المعامل المطلوب إدخاله فى الجدول. وبالنسبة لمعاملات المقارنة العددية فهى معروفة ولاتحتاج إلى شرح. أما معاملات المقارنة الحرفية فيتم شرحها فى الجدول التالى :

المعامل	الوظيفة
= Matches	وهو يعنى أن الحروف الموجودة فى الحقل تماثل تماما الحروف الموجودة فى الثابت الحرفى المستخدم فى المقارنة متضمنا حالة الحروف إذا كانت كبيرة (Upercase) أو صغيرة (Lowercase).
< > Does not match	وهو يعنى أن الحروف فى الخقل لاتماثل الحروف الموجودة فى الثابت الحرفى. وفى هذه الحالة يتم اختيار السجلات التى لايطابق حقل معين فيها مقدارا ثابتا معينا (Constant). فمثلا إذا أريد استرجاع كل السجلات الخاصة بالموظفين من جنسيات غير مصرية مثلا يتم إدخال الشرط كالآتى : Nation < > Egypt
= Begins With	وهو يعنى أن أول حرف أو مجموعة من الحروف فى حقل معين تماثل الحرف أو الحروف المطلوب مقارنتها.
End With	وهو يعنى أن آخر حرف أومجموعة من الحروف فى حقل معين تماثل الحرف أو الحروف المطلوب مقارنتها.
$ Contains	وهو يعنى أن الحقل يحتوى على حروف معينة (فى أى مكان داخله). فمثلا يمكن البحث عن السجلات الخاصة بالموظفين الذين يسكنون بحى شبرا وذلك عن طريق البحث عن كلمة شبرا فى حقل العنوان (Address).
Does not contain	وهو يعنى أن الحقل لايحتوى داخله على حروف معينة. وهو عكس المعامل السابق.
Is contained in	وهو يعنى أن الحقل الحرفى موجود ضمن مجموعة معينة من الحروف.
Is not contained in	وهو يعنى أن الحقل الحرفى غير موجود ضمن مجموعة معينة من الحروف.
> Comes after	وهو يعنى أن أول حرف فى الحقل الحرفى يأتى بعد حرف معين فى الترتيب الهجائى.
> = Comes after or matches	وهو يعنى أن أول حرف فى الحقل الحرفى يماثل أو يأتى بعد حرف معين فى الترتيب الهجائى.

الفصل التاسع عشر

التقارير والعناوين البريدية

(Reports and Labels)



يحتاج المستخدم إلى تقارير مكتوبة متضمنة بيانات من بعض السجلات كما يحتاج فى بعض الأحيان إلى عناوين بريدية (Labels) تحتوى على بيانات سريعة من أى سجل مثل الإسم والعنوان والتليفون. ولتنفيذ ذلك يلزم أولا إنشاء ملف التقرير أو العناوين البريدية واستخدامه بعد ذلك فى كتابة بيانات أى سجل أو مجموعة من السجلات.

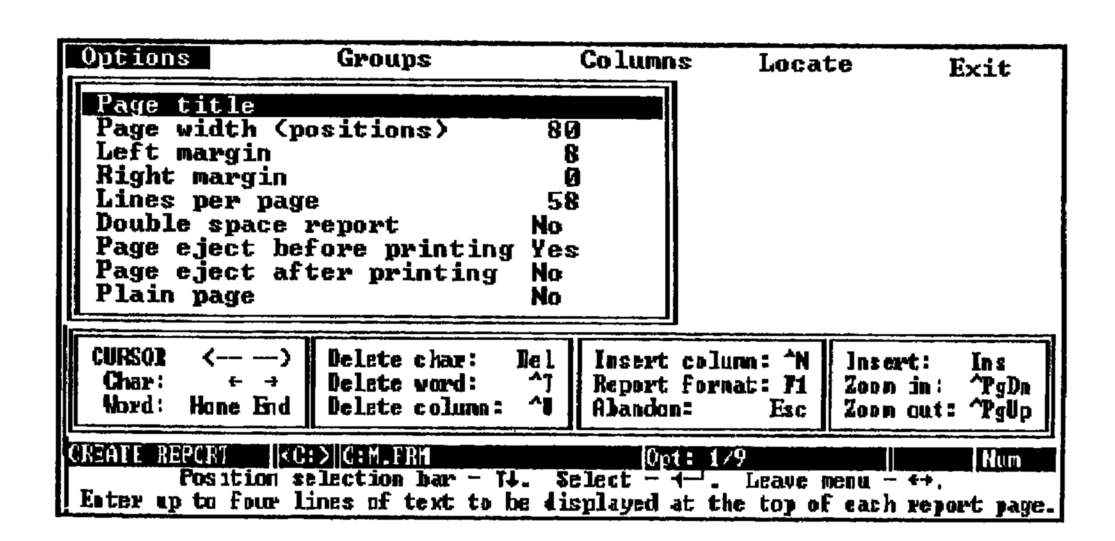
١٩ - ١ إنشاء ملف التقرير (Repoart File)

قبل البدء فى إنشاء ملف التقرير يجب أولا التاكد من فتح ملف قاعدة البياناب المستخدم (Database File) حتى يتم اختيار الحقول المطلوب ظهورها فى التقرير. وكما سبق القول فى إنشاء ملفات البحث (Query Files) يمكن إنشاء عدة ملفات تقارير (Report Files) ثم اختيار ملف التقرير المطلوب استخدامه وقت الحاجة.

ولإنشاء ملف التقرير يتم اتباع الخطوات التالية:

١ - يتم فتح قائمة الإنشاء (Create) واختيار (Report).
٢ - يتم اختيار وحدة الأقراص المطلوب تخزين الملف فيها.
٣ - يتم كتابة إسم ملف التقرير المطلوب إنشاؤه مع ملاحظة أن البرنامج يضيف الإمتداد (FRM.).
٤ - يلاحظ ظهور عمود الإختيارات (Menu Bar) مع وقوف مؤشر هذا العمود على أول اختيار وهو الإختيار (Options) مع فتح القائمة الخاصة به.
٥ - يلاحظ وجود عدة اختيارات خاصة بعنوان صفحة التقرير (Page Title) وأبعاد الصفحة وشكل الكتابة ... وهكذا.
٦ - يلاحظ وجود قائمة مساعدة (Help) أسفل الشاشة لمساعدة المستخدم عند كتابة عنوان التقرير والبيانات المختلفة. وذلك عن طريق توضيح وظائف الأسهم المختلفة التى يتم عن طريقها تحريك مؤشر الكتابة. ولإخفاء هذه القائمة يتم الضغط على مفتاح (F1) ، كما يمكن الضغط عليه مرة ثانية لعرض قائمة المساعدة (Help) عند الحاجة إلى ذلك. ويلاحظ أيضا ظهور مساحة خالية مكان هذه القائمة تسمى (Report Format) تظهر بالتبادل مع قائمة المساعدة عند الضغط على مفتاح (F1).

انظر الشكل (١٩ - ١)



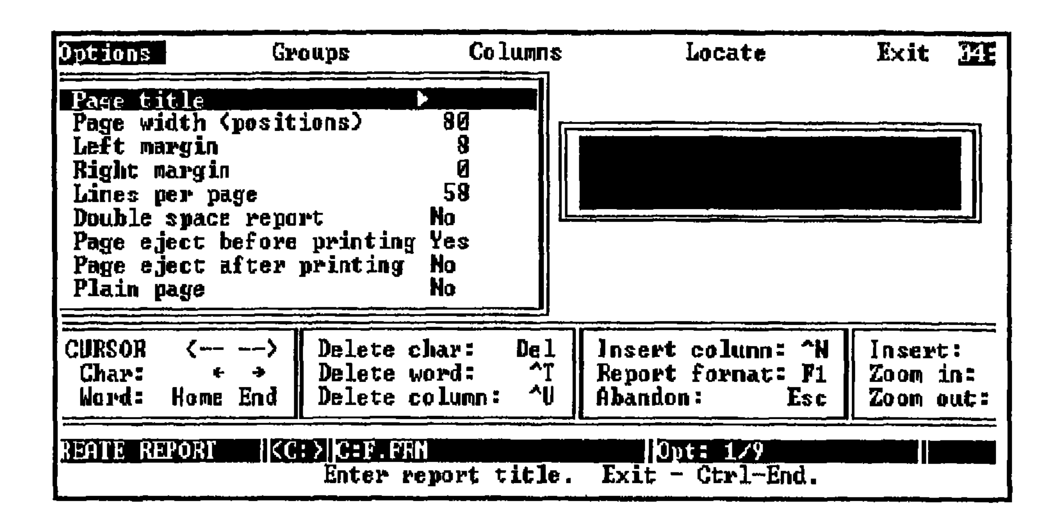
شكل (١٩ - ١)

١٩ - ١ - ١ عنوان التقرير

يسمح البرنامج بكتابة عنوان للتقرير حتى أربعة سطور ويتم ذلك عن طريق الآتى :

مع وجود العمود الضوئى (Highlight) عند الإختيار (Page Title) يتم الضغط على مفتاح الإدخال. ويلاحظ ظهور مستطيل يمكن عن طريقه إدخال العنوان المطلوب ويظهر مؤشر صغير ليساعد على الكتابة. وعند الإنتهاء من كتابة عنوان التقرير يتم الضغط على مفتاح الإدخال عدة مرات حتى يصل المؤشر إلى السطر الأخير ثم يتم الضغط على مفتاح الإدخال مرة أخرى حتى يتم إدخال العنوان.

أنظر الشكل (١٩ - ٢)



شكل (١٩ - ٢)

١٩ - ١ - ٢ التحكم فى شكل الصفحة (Page Format)

تستخدم باقى الإختيارات فى قائمة (Options) فى تحديد شكل الصفحة ، مع ملاحظة أن القيم الموجودة أمام كل اختيار هى القيم المبدئية (Default) وهى تكون فى معظم الأحيان مناسبة ولاتحتاج إلى تغيير. ولتغيير أى اختيار يتم تحريك العمود الضوئى (Highlight) إلى هذا الإختيار والضغط على مفتاح الإدخال ثم كتابة الرقم الجديد. أو يتم الضغط على مفتاحى (| |) لزيادة الرقم المكتوب أو إنقاصه بمقدار (١) مع كل ضغطة.

وفى الجزء التالى يتم توضيح اختيارات هذه القائمة والقيم المبدئية (Default) الخاصة بكل اختيار.

الإختيار	الشرح
Title	وهو عبارة عن أربعة سطور يتم كتابتها فوق كل صفحة من التقرير كعنوان لهذا التقرير.

الإختيار	الشرح
Page Width	وهو أكبر عدد من الحروف يمكن كتابته فى السطر الواحد والقيمة المبدئية ٨٠ حرفا والمدى من ١ إلى ٥٠٠ حرف.
Left Margin	وهو طول المسافة المحصورة بين حرف الصفحة الأيسر وأول حرف مطبوع والقيمة المبدئية ٨ والمدى من صفر حتى عرض الصفحة.
Lines per page	وهو أكبر عدد من السطور يمكن طباعته فى الصفحة. والعدد المبدئى (Default) هو ٥٨ سطرا والمدى من ٣٠ إلى ١٠٠ سطر.
Double space report	وهو يسمح بترك سطر خال بين كل سطرين متتاليين. والوضع المبدئى يكون (Single-spacing) أى بدون سطور خالية.
Page eject before printing	وهو يؤدى إلى تحريك الورقة إلى بداية الصفحة التالية عند بدء الطباعة والوضع المبدئى (Yes).
Page eject after printing	وهو يؤدى إلى تحريك ورقة خالية فى نهاية الطباعة والوضع المبدئى (No).
Plain page	وهو يعنى أن التقرير لايحتوى على أرقام الصفحات والتاريخ على الصفحات والوضع المبدئى هو (No) أى كتابة هذه البيانات. وعند تغييره إلى (Yes) لايتم كتابة هذه البيانات.

١٩ - ١ - ٣ تجميع أو تصنيف السجلات (Grouping)

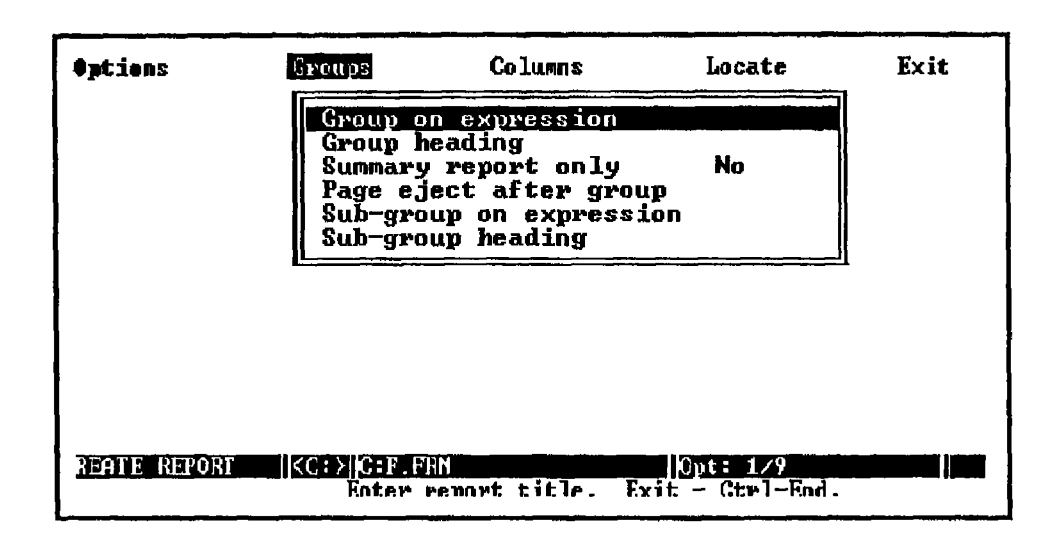
وهو الإختيار الثانى فى عمود الإختيارات (Menu Bar) ويتم عن طريقه تجميع أو تصنيف السجلات تبعا للحقل الفهرسى (Index Field) إلى مجموعات رئيسية ومجموعات فرعية. حيث يتم وضع كل مجموعة من السجلات التى تشترك فى الحقل الفهرسى مع بعضها مع تحديد عنوان لهذه المجموعة. فمثلا إذا كان الحقل الفهرسى

هو حقل تاريخ الإلتحاق (Dat_Ent) فيمكن أن تظهر بيانات الطلبة الذين التحقوا فى كل عام فى مجموعة منفصلة.

وهذا الإختيار لايتم اختياره إلا فى الحالات التى تتطلب ذلك حيث يمكن للمستخدم عدم الدخول فى قائمة التجميع عند إنشاء التقرير.

ويتم استخدام هذه القائمة باتباع الخطوات التالية :

١ - يتم نقل مؤشر عمود الإختيارات إلى الإختيار (Groups) فيتم فتح القائمة الخاصة به. انظر الشكل (١٩ - ٣)



شكل (١٩ - ٣)

٢ - يتم اختيار (Group On Expression) وهو يعنى تحديد الحقل المطلوب التجميع بناء عليه.

٣ - يتم الضغط على (F10) لعرض أسماء الحقول واختيار إسم الحقل المطلوب إدخاله ثم الضغط على مفتاح الإدخال.

٤ - يتم تحريك العمود الضوئى (Highlight) إلى (Group Heading) وكتابة الإسم المطلوب لهذه المجموعة من السجلات ثم الضغط على مفتاح الإدخال.

٥ - يتم الضغط على مفتاح (<--) للخروج من قائمة التجميع (Groups). وفيما يلى جدول يوضح الإختيارات الخاصة بالتجميع وشرح كل منها.

الإختيار	الشـرح
Group on expression	ويتم عن طريقه كتابة إسم الحقل أو العلاقة التى يتم التجميع أو التصنيف بناء عليها. حيث يمكن التجميع بناء على علاقة بين حقول معينة.
Group heading	ويتم عن طريقه إعطاء عنوان للمجموعة ويمكن إدخال حتى ٤ سطور كعنوان.
Summary report only	وهو يؤدى إلى طباعة معلومات مختصرة عن السجلات الموجودة فى المجموعة.
Page eject after group	وهو يؤدى إلى طباعة كل مجموعة من السجلات فى صفحة منفصلة.
Sub-group on expression	وهو يؤدى إلى إنشاء مجموعات فرعية تبعا لحقل معين.
Sub_group heading	وهو يؤدى إلى كتابة عنوان للمجموعة الفرعية عند الطباعة.

١٩ - ١ - ٤ تخطيط الأعمدة (Column Layout)

والمقصود به تحديد مكان وعرض كل عمود يمثل حقلا معينا من حقول الملف. ولتنفيذ ذلك يتم تحديد مكان كل حقل والعنوان الخاص به وذلك كالآتى :

١ - يتم تحريك مؤشر عمود القوائم (Menu Bar) إلى الإختيار (Column) فيتم فتح القائمة الخاصة به. انظر الشكل (١٩ - ٤).

٢ - يلاحظ وقوف العمود الضوئى (Highlight) على الإختيار (Contents). ويمكن كتابة إسم الحقل المراد إختياره أو يتم الضغط على مفتاح (F10) لعرض قائمة الحقول والإختيار منها.

٣ - يتم تحريك العمود الضوئى إلى الإختيار (Heading) ثم كتابة العنوان المراد طباعته لهذا الحقل. ويمكن كتابة عنوان مختلف عن إسم الحقل لأنه لايكون هناك شروط محددة لعدد حروف هذا العنوان. وذلك عكس اسم الحقل يكون مقيدا بالشروط المعروفة. كما يمكن كتابة هذا العنوان ...

ثلاثة حسب الحاجة. كما يمكن ترك سطور خالية قبل هذا العنوان وذلك بالضغط على مفتاح الإدخال قبل بدء الكتابة عددا من المرات يقابل عدد السطور المراد تركها خالية. انظر الشكل (١٩ - ٥)

```
                          Contents
                          Heading
                          Width                  0
                          Decimal places
                          Total this column

 Report Format
>>>>>>>>

CREATE REPORT    ||<C:>||C:F.FRM             ||Column: 1
   Position selection bar - ↑↓.  Select - ←┘.  Prev/Next
     Enter a field or expression to display in the indicat
```

شكل (١٩ - ٤)

```
Options     Groups     Columns     Locate      Exit
                          Contents             NAME
                          Heading              NAME
                          Width                30
                          Decimal places
                          Total this column
 Report Format
>>>>>>>>NAME

        XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
CREATE REPORT    ||<C:>||C:F.FRM             ||Column: 1
   Position selection bar - ↑↓.  Select - ←┘.  Prev/Next
     Enter up to four lines of text to display above the
```

شكل (١٩ - ٥)

٤ - يلاحظ ظهور العدد الممثل للعرض (Width) الذى سبق تحديده لهذا الحقل عند بداية إنشاء ملف قاعدة البيانات.

٥ - يلاحظ أيضا ظهور عنوان الحقل (Heading) وتحته علامات (X) بقدر عرض الحقل (Width) الذى سبق تحديده وذلك فى المستطيل الموجود أسفل الشاشة والذى يستخدم فى تحديد شكل التقرير.

٦ - يتم الضغط على مفتاح (PgDn) لتوصيف عمود آخر فى التقرير فيلاحظ ظهور قائمة خالية يتم من خلالها تحديد الحقل الثانى المطلوب تمثيله فى التقرير.

٧ - يتم إدخال باقى الحقول المطلوب عرضها فى التقرير بنفس الطريقة.

٨ - يتم الضغط على مفتاح السهم يمين (-->) للخروج من قائمة الأعمدة (Columns) ويلاحظ ظهور الشكل النهائى للتقرير على الشاشة.

ملاحظات

١ - إذا كان الحقل حرفيا تظهر الحروف (X) بعرض الحقل الذى سبق تحديده. وإذا كان الحقل عدديا تظهر الأرقام (٩) بعرض الحقل أيضا.

٢ - عند زيادة عرض عنوان الحقل (Heading) عن عرض الحقل (Width) فإن البرنامج يقوم بضبط عرض العمود حتى يغطى عرض عنوان الحقل.

٣ - يلاحظ وجود الإختيار(Total this column) فى قائمة الأعمدة(Columns). فإذا أريد تجميع الأعداد الموجودة فى هذا الحقل يتم تعديل الوضع المبدئى لهذا الإختيار من (No) إلى (Yes).

١٩ - ١ - ٥ إختبار الحقول قبل تخزين الملف

يمكن الرجوع إلى أى حقل وتعديل بياناته عن طريق قائمة (Locate). ولتنفيذ ذلك يتم اتباع الخطوات الآتية :

١ - يتم فتح قائمة (Locate) ويلاحظ ظهور أسماء الحقول الموجودة بالتقرير.

٢ - يتم تحريك المؤشر الضوئى (Highlight) لاختيار الحقل المطلوب إختباره والضغط على مفتاح الإدخال.

٣ - يلاحظ ظهور القائمة الخاصة بهذا الحقل متضمنة إسم الحقل وعنوانه فى التقرير وعرضه ويمكن تعديل هذه البيانات للوصول إلى شكل التقرير المطلوب.

١٩ - ١ - ٦ تخزين وتعديل التقرير

لتخزين التقرير يتم تحريك مؤشر عمود القوائم (Menu Bar) إلى قائمة الخروج (Exit) ثم اختيار الأمر (SAVE).

ولتعديل التقرير يتم اختيار قائمة التعديل (Modify) من القوائم الرئيسية الثمانية التى سبق ذكرها. ويتم اختيار التقرير(Report) من هذه القائمة فيلاحظ ظهور قائمة بأسماء ملفات التقارير التى سبق تخزينها فيتم اختيار التقرير المطلوب تعديله. ويلاحظ ظهور نفس القوائم المستخدمة فى إنشاء التقرير.

١٩ - ١ - ٧ طباعة التقرير

يتم استخدام ملف التقارير فى عرض بيانات سجلات محددة على الشاشة أو طباعتها على الطابعة. ولتنفيذ ذلك يتم اتباع الخطوات التالية :

١- يتم التأكد أولا من فتح ملف قاعدة البيانات المطلوب كما يتم فتح ملف الفهرس (Index File).
٢- يتم فتح قائمة الإسترجاع (Retrieve) واختيار (Report).
٣- يتم اختيار وحدة الأقراص التى تحتوى على القرص المخزن به ملف التقارير المطلوب ثم اختيار الملف المطلوب.
٤- يتم تحديد شروط البحث (Search Conditions) ومجال البحث (Search Scope) لاختيار سجلات محددة حسب الحاجة.
٥- يتم اختيار (Execute the command) فيظهر السؤال التالى :

Direct the output to the printer? (Y/N)

٦- يتم التأكد من أن الطابعة جاهزة ثم كتابة (Y) فيتم طباعة التقرير.
٧- إذا أريد عرض التقرير على الشاشة فقط يتم كتابة (N) أمام السؤال السابق.

١٩ - ٢ إنشاء العناوين البريدية (Labels)

فى بعض الأحيان يكون مطلوبا طباعة أو عرض عناوين مختصرة وسريعة (Labels). وهذه العناوين تتضمن بعض البيانات الضرورية مثل الإسم والعنوان ورقم التليفون مثلا. وتتبع فى إنشاء ملف العناوين وتعديله نفس الخطوات السابق شرحها فى إنشاء وتعديل التقرير.

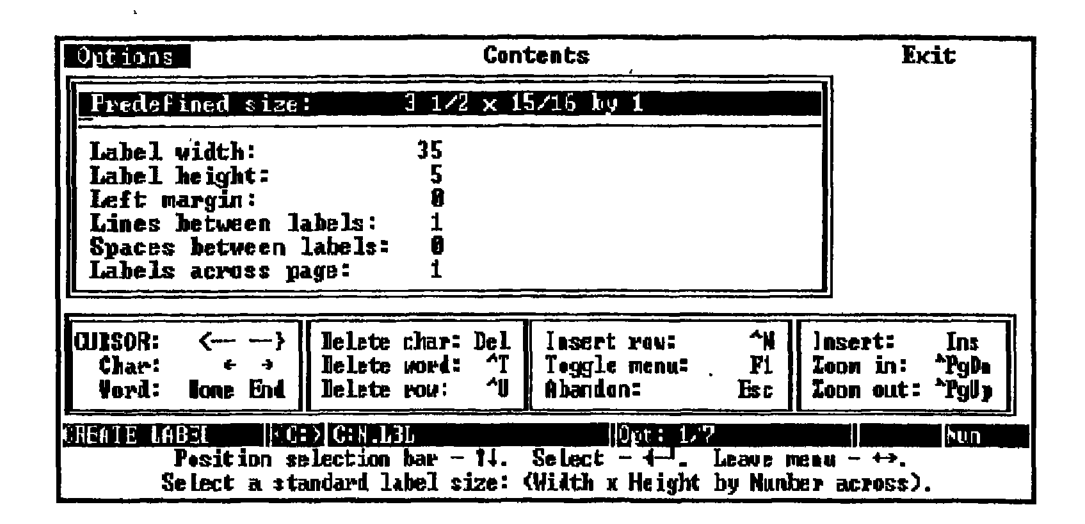
ولإنشاء ملف العناوين البريدية (Labels) ، يتم اتباع الخطوات التالية :

١ - يتم فتح قائمة التجهيز (Set Up) واختيار ملف قاعدة البيانات (Data Base).
٢ - يتم اختيار ملف الفهرس المطلوب فتحه.
٣ - يتم فتح قائمة الإنشاء (Create) واختيار (Label).
٤ - يتم اختيار وحدة الأقراص التى تحتوى على القرص المطلوب تخزين الملف عليه.
٥ - يتم كتابة إسم ملف العناوين المطلوب إنشاؤه والضغط على مفتاح الإدخال.
٦ - يلاحظ فتح قائمة العناوين مع وقوف المؤشر العلوى على قائمة (Options) وبالتالى يتم فتح القائمة الفرعية الخاصة بها. وهذه القائمة تحتوى على اختيارات يتم عن طريقها تحديد أبعاد الطباعة.

١٩ - ٢ - ١ تحديد أبعاد الصورة المطبوعة

لتحديد أبعاد الطباعة يتيح البرنامج ثلاثة أبعاد قياسية (Standard) ويتم ذلك باتباع الخطوات التالية :

١ - مع وجود العمود الضوئى (Highlight) على (Predefined Size) يتم الضغط على مفتاح الإدخال فيلاحظ ظهور أرقام تمثل الأبعاد القياسية للصورة المطبوعة مثل (3 1/2 X 15/16) by 3) ومع كل ضغطة على مفتاح الإدخال تظهر أبعاد قياسية جديدة. والرقم السابق يؤدى إلى طباعة تقرير أبعاده (١.٢ ٣ X ١٥.١٦ بوصة) وطباعة ثلاثة تقارير فى الصفحة الواحدة. أنظر الشكل (١٩ - ٦).
٢ - يمكن تعديل باقى الإختيارات الخاصة بأبعاد التقرير المختصر كما يمكن ترك القيم المبدئية (Default) الموجودة فى الجدول كما هى حيث أنها تعتبر مناسبة.



شكل (١٩ - ٦)

والجدول التالى يوضح اختيارات هذه القائمة والقيم المبدئية (Default) لها.

الإختيار	الشرح
Label width	وهو أكبر عدد من الحروف فى السطر الواحد من التقرير والمدى من ١ إلى ١٢٠ حرفا.
Line Hight	وهو يمثل عدد السطور فى التقرير الواحد والمدى من ١ الى ١٦ سطرا.
Left margin	وهو يمثل المسافة بين الحرف الأيسر للورقة وأول حرف مطبوع والمدى من صفر إلى ٢٥٠ حرفا.
Lines between labels	وهو يمثل المسافة الرأسية بالسطور بين سطور التقرير والمدى من صفر إلى ١٦ سطرا.
Spaces between labels	وهو يمثل المسافة الأفقية بين التقارير والمدى من صفر إلى ١٢٠ حرفا.
Labels across page	وهو يمثل عدد التقارير المطبوعة فى الصفحة والمدى من ١ الى ١٥ تقريرا.

ملاحظة

هذه القيم المبدئية تتغير بتغير الأبعاد القياسية للتقرير (Predefined Sizes).

١٩ - ٢ - ٢ إدخال محتويات التقرير

يتم إدخال محتويات تقرير العناوين البريدية (Label) عن طريق الخطوات التالية :

١ - يتم تحريك المؤشر العلوى إلى الإختيار (Contents) فيتم فتح القائمة الخاصة به.
٢ - يتم تحريك العمود الضوئى الخاص بهذه القائمة إلى السطر الثانى والضغط على مفتاح الإدخال فيلاحظ ظهور العلامة < وظهور مؤشر صغير على هذا السطر يتم عن طريقه كتابة الحقل المطلوب إدخاله.
٣ - يمكن كتابة أسماء الحقول أو الضغط على مفتاح (F10) لإظهار القائمة الخاصة بها واختيار الحقل المطلوب.
٤ - يمكن إدخال أكثر من حقل فى نفس السطر عن طريق كتابة العلامة (,) بين أسماء الحقول.

ملاحظة

استخدام علامة (,) بين أسماء الحقول يؤدى إلى التخلص من المسافات الزائدة فى نهاية الحقل (Trimming). أما إذا أريد الإحتفاظ بهذه المسافات بين الحقول فتستخدم علامة الجمع (+) بدلا من الفاصلة (,).

٥ - يتم الضغط على مفتاح الإدخال فيتم إدخال هذا السطر والإنتقال إلى السطر التالى.
٦ - يتم إدخال باقى السطور بنفس الطريقة.
٧ - لتخزين تقرير العناوين البريدية (Label) يتم تحريك المؤشر العلوى إلى آخر اختيار وهو (Exit) ثم اختيار (Save).

١٩ - ٢ - ٣ طباعة تقارير العناوين البريدية

لطباعة تقارير العناوين البريدية لسجل معين أو لمجموعة من السجلات يتم اتباع الخطوات التالية :

١ - يتم فتح قائمة الإسترجاع (Retrieve) واختيار (Label).
٢ - يتم تحديد وحدة الأقراص التى تحتوى على القرص المخزن عليه ملف التقارير المختصرة الذى سبق إنشاؤه.
٣ - يتم تحديد شروط البحث ومدى البحث كما سبق الإيضاح.
٤ - يتم اختيار (Execute the command) فيظهر السؤال الآتى على الشاشة :

Direct the output to the printer ? (Y / N)

٥ - يتم التأكد من توصيل الطابعة وتشغيلها ثم كتابة (Y).

١٩ - ٣ تلخيص البيانات (Summarizing Data)

توجد ثلاثة اختيارات فى قائمة الإسترجاع (Retrieve) تؤدى إلى تجميع البيانات العددية فى السجلات التى يتم اختيارها. وهذه الإختيارات هى (Sum) ، (Average) ، (Count).

والإختياران (Sum) ، (Average) يعملان على الحقول العددية فقط حيث يؤدى الإختيار (Sum) إلى تجميع الحقول العددية فى الملف. كما يؤدى الإختيار (Average) إلى حساب المتوسطات العددية للحقول العددية فى الملف. والإختيار (Count) يحسب عدد السجلات التى تحقق شرطا أو شروطا معينة. أنظر الشكل (١٩ - ٧).

```
Set Up Create Update Position Retrieve Organize ModifyTools
                                  List
                                  Display
                                  Report
                                  Label
                                  ─────
                                  Sum
                                  Average
                                  Count

ASSIST              ||<C:>||M                    ||Rec: 1/4
Move selection bar - ↑↓. Select - ◄─┘.  Leave menu - ↔. He
           Display the totals of the specified numeric f
```

شكل (١٩ - ٧)

الفصل العشرون

ربط قواعد البيانات

(Relating Databases)



عندما تكون قاعدة البيانات كبيرة - أى تحتوى على عدد كبير من الحقول وعدد كبير من السجلات - فالأفضل فى هذه الحالة إنشاء عدة ملفات بدلا من ملف واحد. وذلك لأن الملف الكبير له عيوب كثيرة مثل الآتى :

١ - عند البحث عن بيان خلال الملف يستغرق البحث وقتا طويلا.
٢ - يحتل الملف جزءا كبيرا من الذاكرة المؤقتة عند تحميله وهذا يؤثر على سرعة تشغيل البيانات.

وعند تقسيم قاعدة البيانات على عدة ملفات يجب أن تكون كل الملفات محتوية على حقل مشترك وهذا الحقل المشترك يجب أن يكون منفردا (Unique) وعن طريق هذا الحقل يمكن ربط الملفات ببعضها. وهذا يتيح للمستخدم الإسترجاع السريع لأى سجل وتعديل البيانات المطلوب تعديلها. كما أن أى تعديل فى حقل معين فى أى ملف يؤثر فى أى حقول معتمدة على هذا الحقل فى الملفات الأخرى.

ويمكن أيضا استخدام ملف المنظر (View File). وهو عبارة عن ملف يتم فيه تخزين حقول من عدة ملفات مرتبطة ببعضها بواسطة حقل مشترك. ويمكن استرجاع هذا الملف فى أى وقت واسترجاع البيانات المطلوبة والتى يقوم هذا الملف بتجميعها من الملفات المرتبطة به. انظر الشكل (٢٠ - ١)

```
Set Up          Relate        Set Fields        Options        Exit
 >CADETS.DBF
  MOS.DBF         NAME.NDX
  HISTORY.DBF

CREATE VIEW  |<A:>|A:M.VUE                     |Opt: 1/1
 Position selection bar - ↑↓. Select - ◄─┘. Close files - Esc.
  Select up to seven index files. The first file selected is th
```

شكل (٢٠ - ١)

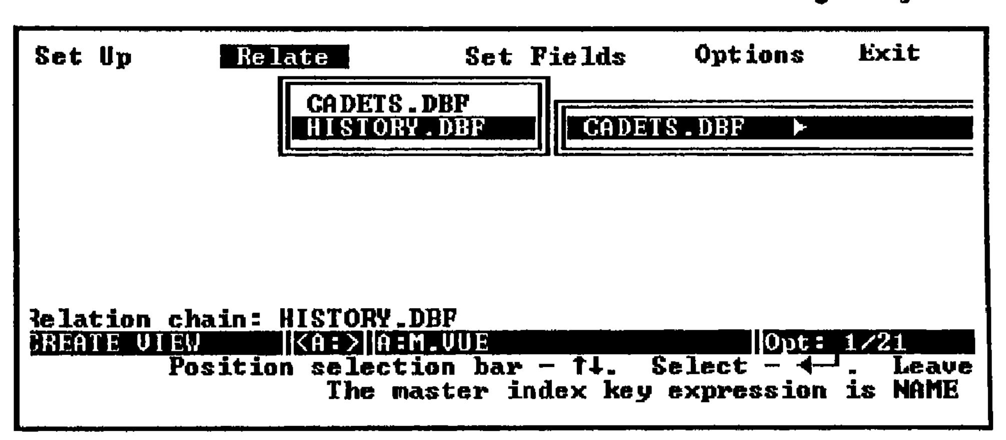
ملاحظة

لايستخدم ملف المنظر (View File) فى إضافة (Append) سجلات جديدة ولكن يتم إضافة السجلات الجديدة عن طريق الملفات الأصلية.

٢٠ - ١ إنشاء ملف المنظر (View File)

لإنشاء ملف المنظر يتم أولا تحديد الملفات المطلوب إدخالها فيه. كما يتم تحديد الحقول المطلوبة من كل ملف والمطلوب إدخالها فى ملف المنظر (View File). ولإنشاء هذا الملف تتبع الخطوات التالية :

١ - يتم فتح قائمة الإنشاء (Create) واختيار (View).
٢ - يتم اختيار وحدة الأقراص التى يراد تخزين الملف فيها.
٣ - يتم كتابة إسم ملف المنظر المطلوب إنشاؤه ويلاحظ ظهور عمود القوائم(Menu Bar) مع وقوف المؤشر الخاص به على الإختيار (Set Up). وبالتالى يتم فتح القائمة الخاصة به والتى تحتوى على ملفات قواعد البيانات الموجودة على القرص. فيتم اختيار الملف الأول وكذلك اختيار ملف الفهرس الخاص به (Index File) ثم الضغط على مفتاح السهم يمين (<--) للرجوع إلى قائمة ملفات قواعد البيانات واختيار الملف الثانى وهكذا.
٤ - يتم تحريك المؤشر العلوى إلى الإختيار (Relate) فيلاحظ فتح القائمة الخاصة بهذا الإختيار والتى تحتوى على أسماء الملفات التى تم اختيارها لإدخالها فى ملف المنظر. أنظر الشكل (٢٠ - ٢)



شكل (٢٠ - ٢)

٥ - مع وجود العمود الضوئى (Highlight) على أول ملف يتم الضغط على مفتاح الإدخال فيلاحظ ظهور قائمة الملفات المرتبطة بهذا الملف.

٦ - يتم اختيار كل ملف من هذه الملفات والضغط على مفتاح (F10) لإظهار قائمة بحقول هذا الملف. ومن هذه القائمة يتم اختيار الحقل المراد استخدامه فى ربط هذا الملف بالملف الأول.

٧ - يعتبر الملف الأول هو الأصل وباقى الملفات مرتبطة بهذا الملف طبقا للحقل الفهرسى. أى أن السجلات فى كل ملف يتم ترتيبها حسب ترتيب الحقل الفهرسى فى الملف الأول.

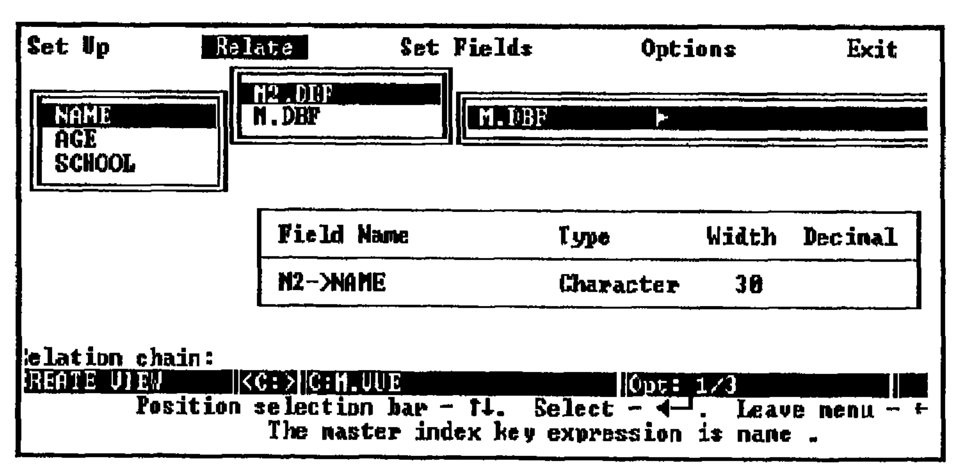
٢٠ - ٢ إختيار حقول ملف المنظر (View File)

بعد ربط الملفات يتم تحديد الحقول المراد اختيارها من كل ملف لإدخالها فى ملف المنظر. ولتنفيذ ذلك تتبع الخطوات التالية :

١ - يتم تحريك المؤشر العلوى إلى الإختيار (Set Fields) فيتم فتح القائمة الخاصة به. انظر الشكل (٢٠ - ٣).

٢ - يلاحظ ظهور قائمة بملفات قواعد البيانات التى تم ربطها بحيث يظهر الملف الأول فى أول القائمة وبعده باقى الملفات.

٣ - يتم تحريك المؤشر الضوئى إلى إسم كل ملف والضغط على مفتاح الإدخال فتظهر قائمة بحقول هذا الملف ويتم اختيار الحقول المراد إدخالها فى ملف المنظر وهكذا.



شكل (٢٠ - ٣)

٢٠ - ٣ تخزين ملف المنظر

لتخزين ملف المنظر (View File) يتم تحريك المؤشر العلوى إلى الإختيار (Exit) والضغط على مفتاح الإدخال ويتم اختيار (Save).

وعندما يراد تعديل الملف بعد ذلك يتم الدخول إلى قائمة التعديل (Modify) من القائمة الرئيسية للبرنامج ثم اختيار(View). ويلاحظ فى هذه الحالة ظهور نفس القوائم التى تظهر فى حالة إنشاء ملف منظر جديد.

٢٠ - ٤ فتح ملف المنظر

لفتح ملف المنظر يتم الدخول فى قائمة التجهيز (Set Up) واختيار (View) وكتابة إسم الملف. ويتم استخدامه بعد ذلك فى عرض وتعديل البيانات فى الحقول التى تم اختيارها من الملفات المختلفة.

تحـذير

يراعى عند إجراء تعديل فى بيانات الحقول عدم تعديل بيانات الحقل المشترك الذى سبق تحديده.

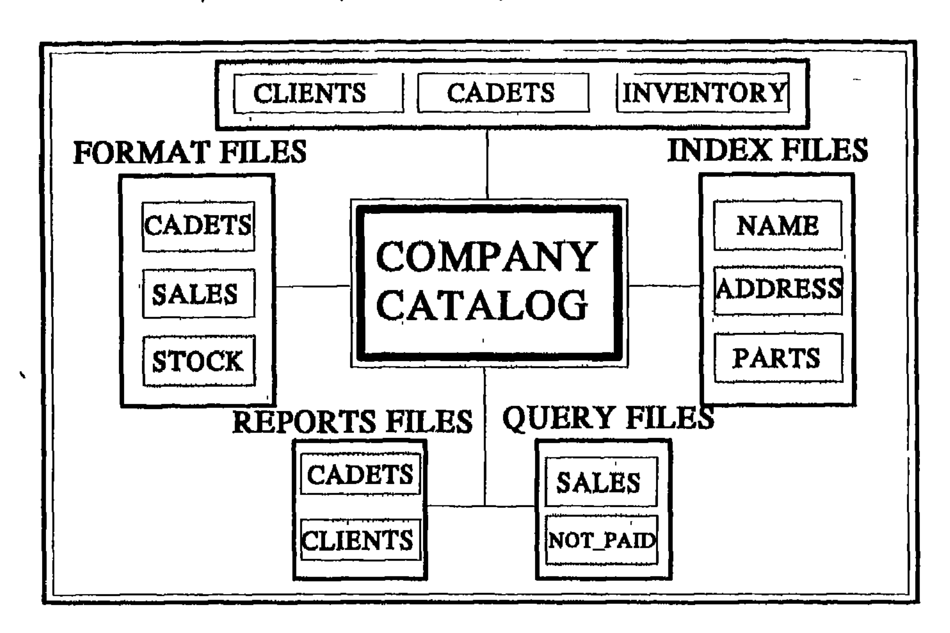
٢٠ - ٥ إستخدام الكتالوجات

كما يستخدم نظام التشغيل (MS-DOS) نظام الفهارس (Directories) والفهارس الفرعية (Subdirectories) فى تنظيم الملفات على القرص. فإن برنامج (DBase III+) يستخدم الكتالوجات (Catalogs) فى تنظيم الملفات الخاصة بقواعد البيانات (Database Files) والملفات المرتبطة بها مثل ملفات الفهرس (Index Files) وملفات البحث (Query Files) وملفات التقارير (Report Files) و ... وهكذا.

انظر الشكل (٢٠ - ٤)

وتفيد هذه العملية عندما يكون عدد الملفات المخزنة على القرص الصلب(Hard Disk) كبيرا جدا حيث يمكن إدخال كل مجموعة من قواعد البيانات التى تؤدى وظائف متقاربة فى كتالوج منفصل. وعندما يراد استخدام الملفات الخاصة بقاعدة البيانات مثل ملفات

التقارير أو ملفات البحث ، أو .. الخ فإن قائمة الملفات التى تظهر للإختيار منها لاتحتوى إلا على الملفات الموجودة فى هذا الكتالوج وبالتالى لايتم عرض قوائم كبيرة للملفات.



شكل (٢٠ - ٤)

وعند فتح الكتالوج فإن أى ملفات جديدة يتم إنشاؤها تضاف إلى هذا الكتالوج. وإنشاء الكتالوج لايتم من خلال برنامج المساعد (Assistant) ولكن يتم من خلال أوامر النقطة (Dot Commands) التى سيتم شرحها فيما بعد.

أما فتح الكتالوج فيمكن أن يتم من خلال برنامج المساعد (Assistant) عن طريق قائمة التجهيز (Set Up).

وعندما يراد تغيير الكتالوج الجارى استخدامه (Active Catalog) بكتالوج آخر يتم الرجوع إلى قائمة التجهيز (Set Up) واختيار الكتالوج المطلوب. وفى هذه الحالة يصبح الكتالوج القديم غير مستخدم (Inactive) ويصبح الكتالوج الجديد هو الكتالوج المستخدم.



الفصل الحادي والعشرون

اوامـــر النقـــطة

(Dot Commands)



فى الأجزاء السابقة تم شرح استخدام برنامج المساعد (Assistant) فى إنشاء ملفات قواعد البيانات وفتحها وكذلك إنشاء باقى الملفات المرتبطة بها مثل ملف الفهرس (Index) والتقارير (Reports) والبحث (Query) والمنظر (View) و ... الخ. وكما كان واضحا من خلال هذا الشرح فإن هذه العمليات يتم تنفيذها من خلال قوائم تظهر على الشاشة.

وهناك طريقة أخرى لاتعتمد على القوائم ولكنها تعتمد على كتابة الأوامر مباشرة عند النقطة التى تظهر على الشاشة فوق عمود الحالة (Status Bar). وهذه النقطة تظهر عند الخروج من القائمة الرئيسية باستخدام مفتاح الهروب (Esc).

أنظر الشكل (٢١ - ١)

```
Command Line    ||<C:>||M                    ||Rec: 1/4
                     Enter a dBASE III PLUS command.
```

شكل (٢١ - ١)

ويلاحظ أن عمود الحالة (Status Bar) يبين هذا التحول من المساعد(Assistant) إلى أمر (Command).

ملاحظة

يمكن عن طريق مسح سطر من ملف المواصفات (Config.sys) جعل برنامج (DBase III +) يبدأ من أوامر النقطة (Dot Commands) بدلا من المساعد (Assistant) الذى يظهر عند بداية تشغيل البرنامج. وهذا السطر يكون كالآتى :

COMMAND = ASSIST

ويتم من خلال أوامر النقطة (Dot Commands) إجراء كل العمليات التى سبق شرحها وأداء عمليات إضافية أيضا. وهذه الطريقة فى إدخال الأوامر يتم استخدامها عادة بعد استخدام برنامج المساعد (Assistant) مدة كافية والتعود على أوامر البرنامج. ويجب

ملاحظة أن كل أمر يتم إدخاله بواسطة برنامج المساعد يظهر فوق عمود الحالة (Status Bar) فى نفس الوقت. وهكذا يمكن عن طريق برنامج المساعد معرفة شكل الأمر المقابل (Syntax) عند إجراء أى عملية.

كما أن دراسة أوامر النقطة (Dot Commands) تعتبر أساسية لمن يريد كتابة البرامج بواسطة برنامج (+ DBase III) وبرامج عائلة (DBase) الأخرى.

٢١ - ١ إدخــال الأوامــر

يتم إدخال الأمر بعد النقطة (Dot) مباشرة ثم يتم إضافة أى معاملات (Parameters) مطلوبة لهذا الأمر. والمعامل (Parameter) هو قيمة تساعد على تحديد عمل الأمر. ويمكن إدخال الأوامر بالحروف الصغيرة أو الكبيرة.

ويمكن تصحيح حروف الأمر باستخدام مفاتيح التصحيح المعتادة مثل مفتاح (Backspace) لحذف الحروف ومفاتيح السهم يمين والسهم يسار (<--- , --->) لتحريك المؤشر إلى مكان الحرف وتصحيحه. وبعد الإنتهاء من كتابة الأمر يتم إدخاله بالضغط على مفتاح الإدخال.

ملاحـظة

إذا حدث خطأ فى كتابة الأمر تظهر الرسالة التالية :

Do you want some help ? (Y/N)

وعند كتابة (Y) تظهر قائمة المساعدة (Help).
وعند كتابة (N) تظهر النقطة مرة ثانية لإعادة إدخال الأمر من جديد.

وهناك بعض الأوامر التى لاتحتاج إلى معاملات(Parameters) مثل الأمر (HELP) والأمر (ASSIST) والأمر (SET). فالأمر (HELP) يؤدى إلى ظهور شاشات المساعدة (Help Screens) التى من خلالها يمكن التعرف على شكل كل أمر (Syntax) وخصائصه.

أما الأمر (ASSIST) فإنه يؤدى إلى ظهور القوائم الخاصة ببرنامج المساعد (Assistant) وإجراء العمليات المطلوب تنفيذها من خلال هذا البرنامج.

والأمر (SET) يمكن استخدامه منفردا بدون معاملات وهذا يؤدى إلى ظهور قائمة يمكن من خلالها تغيير مواصفات البرنامج (Configuration) الخاصة بالشاشة ولوحة المفاتيح ... الخ. وعند اتباعها بمعاملات أخرى فإنها تؤدى عملا محددا يتوقف على هذه المعاملات مثل (SET STATUS ON) ، (SET INDEX TO) و ... الخ.

ملاحظة

يمكن أيضا استخدام برنامج المساعد (Assistant) بالضغط على مفتاح (F2).

٢١ - ٢ عــرض التــاريخ (Display History)

يقوم برنامج (+ DBase III) بتخزين آخر أوامر تم إدخالها فى مخزن مؤقت (Buffer) وهذا المخزن يخزن حتى ٢٠ أمرا. وعندما يراد عرض آخر أوامر تم إدخالها يتم الضغط على مفتاح السهم لأعلى (↑). وفى كل مرة يتم الضغط على هذا المفتاح يظهر آخر أمر تم إدخاله.

ويفيد ذلك عندما يراد إدخال بعض الأوامر عدة مرات فيكفى فى هذه الحالة الضغط على مفتاح الإتجاه لأعلى (↑) عدة مرات للوصول إلى الأمر المطلوب إدخاله والضغط على مفتاح الإدخال وهذا يوفر الوقت اللازم لكتابة الأمر من جديد.

٢١ - ٣ تنفيذ عمليات قاعدة البيانات بواسطة الأوامر

كما سبق الإيضاح فإن جميع العمليات السابق شرحها والتى يتم تنفيذها من خلال برنامج المساعد (Assistant) يمكن تنفيذها من خلال الأوامر (Commands). وذلك بالإضافة إلى بعض العمليات الأخرى التى يتم تنفيذها بواسطة الأوامر فقط.

وكتابة الأوامر عادة تكون أسرع من استخدام القوائم خاصة عند استخدام مفتاح السهم لأعلى (↑) لاستخدام الأوامر التى سبق إدخالها. وفى الأجزاء التالية يتم توضيح بعض العمليات التى سبق شرح تنفيذها من خلال برنامج المساعد (Assistant) مع شرح كيفية استخدام أوامر النقطة (Dot Commands) فى تنفيذها.

٢١ - ٣ - ١ إنشاء واستخدام الكتالوجات

يمكن تجميع كل مجموعة من الملفات فى كتالوج منفصل كما سبق الإيضاح. ولإنشاء كتالوج جديد إسمه (Mycat) مثلا يتم تنفيذ الآتى :

١ - يتم كتابة الأمر التالى بعد النقطة (Dot) مباشرة :

SET CATALOG TO Mycat

ويلاحظ ظهور السؤال التالى على الشاشة :

Create a new catalog File ? (Y/N)

٢ - يتم كتابة (Y) فيتم إنشاء ملف الكتالوج بالإسم (Mycat).

ملاحــظة

بعد إنشاء الكتالوج فإن أى ملفات يتم إنشاؤها أو استخدامها تضاف إلى هذا الكتالوج. ويمكن إغلاق هذا الكتالوج باستخدام الأمر

SET CATALOG OFF

كما يمكن فتح كتالوج آخر باستخدام الأمر

SET CATALOG TO

ثم كتابة إسم الكتالوج المطلوب.

٢١ - ٣ - ٢ إنشاء ملف قاعدة البيانات

لإنشاء ملف قاعدة البيانات يستخدم الأمر (CREATE) مع إسم الملف المراد إنشاؤه. فمثلا عندما يراد إنشاء ملف إسمه (Myfile) يتم تنفيذ الآتى :

١ - يتم كتابة الأمر التالى بعد النقطة (Dot) مباشرة

CREATE Myfile

٢ - يلاحظ ظهور الشاشة المبينة فى الشكل (٢١ - ٢) والتى عن طريقها يتم إدخال أسماء الحقول والبيانات الخاصة بعرض الحقل ونوعه و ... الخ.

٣ - يتم الضغط على مفتاحى (Ctrl-End) لتخزين الملف.

٤ - يتم الضغط على مفتاح الإدخال لتأكيد الرغبة فى تخزين الملف.

٥ - يلاحظ ظهور السؤال الآتى على الشاشة.

Input data records now? (Y/N)

٦ - يتم كتابة (N) عند عدم الرغبة فى إدخال بيانات الملف فى هذا الوقت. وإذا أريد إدخال البيانات يتم كتابة (Y) فتظهر شاشة إدخال البيانات كما سبق الإيضاح.

```
                                                        Bytes remaining: 4000
 CURSOR  <-- -->     INSERT          DELETE          Up a field:      ↑
  Char:    ← →     Char:   Ins     Char:    Del     Down a field:    ↓
  Word:  Home End  Field:  ^N      Word:    ^Y      Exit/Save:     ^End
  Pan:     ^← ^→   Help:   F1      Field:   ^U      Abort:          Esc

    Field Name  Type      Width  Dec      Field Name  Type      Width  Dec

 1              Character

CREATE           |<C:>|FG                  |Field: 1/1      |
                          Enter the field name.
Field names begin with a letter and may contain letters, digits and unde
```

شكل (٢١ - ٢)

٢١ - ٣ - ٣ فتح ملف قاعدة البيانات

يتم فتح ملف قاعدة البيانات باستخدام الأمر (USE). فإذا أريد مثلا فتح الملف الذى سبق إنشاؤه (Myfile) يتم كتابة الأمر التالى بعد النقطة :

USE Myfile

وإذا أريد رؤية أسماء الملفات واختيار الملف المطلوب فتحه يستخدم الأمر التالى :

USE ?

وفى هذه الحالة تظهر أسماء جميع ملفات قواعد البيانات الموجودة على القرص أو الفهرس الفرعى المستخدم.

٢١ - ٣ - ٤ تعديل تركيب ملف قاعدة البيانات (Structure)

لتعديل تركيب ملف قاعدة البيانات يستخدم الأمر التالى :

MODIFY STRUCTURE

ويجب أن يكون الملف قد سبق فتحه باستخدام الأمر (USE) كما سبق الإيضاح. وفى هذه الحالة تظهر الشاشة الخاصة بتوصيف الحقول المبينة فى الشكل (٢١ - ٢).

٢١ - ٣ - ٥ إنشاء ملفات شاشة الإدخال (Format File)

كما سبق الإيضاح فإن من المهم تصميم شاشات لإدخال البيانات واضحة وسهلة بالنسبة للشخص القائم بعملية إدخال البيانات. ولإنشاء شاشة إدخال البيانات يستخدم الأمر التالى :

Create Screen

ثم كتابة إسم الملف المطلوب إنشاؤه.

ويلاحظ فى هذه الحالة ظهور القوائم التى سبق شرحها فى الجزء الخاص بتصميم شاشات الإدخال.

٢١ - ٣ - ٦ فتح ملفات شاشة الإدخال

يتم فتح ملفات شاشة الإدخال باستخدام الأمر :

SET FORMAT TO

ثم كتابة إسم الملف المراد فتحه.

ويمكن عرض أسماء ملفات شاشة الإدخال الموجودة على القرص باستخدام الأمر :

SET FORMAT TO ?

وتظهر فى هذه الحالة قائمة بكل ملفات شاشة الإدخال المخزنة على القرص.

ويمكن إغلاق ملف شاشة الإدخال المفتوح باستخدام الأمر التالى :

SET FORMAT TO

دون كتابة أسماء أى ملفات بعده.

٢١ - ٣ - ٧ إستخدام الأمر (BROWSE)

يستخدم الأمر (BROWSE) كما سبق الإيضاح فى عرض شاشة موضحا بها بيانات مجموعة من السجلات على هيئة أعمدة تمثل الحقول بحيث يكون كل سجل فى سطر. وعن طريق هذه الشاشة يمكن تعديل بيانات أى سجل أو إضافة سجلات جديدة. ولتنفيذ ذلك عن طريق أمر النقطة (Dot Command) يتم كتابة الأمر كالآتى :

BROWSE FIELDS

ثم كتابة أسماء الحقول المراد عرضها على الشاشة. فإذا أريد مثلا عرض حقول الإسم (Name) والعنوان (Address) والوظيفة (Job) يتم كتابة الأمر التالى :

BROWSE FIELDS name , address , job

أنظر الشكل (٢١ - ٣).

```
CURSOR  <-- -->  |            UP   DOWN |    DELETE      | Insert Mode:    Ins
 Char:      ←  → | Record:    ↑     ↓   | Char:     Del  | Exit:          ^End
 Field:Home End  | Page:   PgUp  PgDn   | Field:    ^Y   | Abort:          Esc
 Pan:      ^← ^→ | Help:      F1        | Record:   ^U   | Set Options:  ^Home
NAME-------------------------------- ADDRESS----------------------------- PHONE----
mohamed hasan fathy                  12-ain shams                        36526756
ahmed soliman tarek                  40-ahram-street                     6789889
WALAA MOSTAFA                        NASR CITY                           6394588
HAYTHAM MOSTAFA                      NASR CITY                           7428953

BROWSE          |<C:>|M                        |Rec : 1/4              |

                        View and edit fields.
```

شكل (٢١ - ٣)

٢١ - ٣ - ٨ إستخدام الأمر (GOTO)

عن طريق الأمر (GOTO) يمكن تحريك مؤشر البرنامج إلى سجل محدد وذلك حتى يمكن تعديل بيانات سجل معين. أو عرضها حسب الحاجة. ولاستخدام الأمر (GOTO) فى الوصول إلى السجل رقم ٣ مثلا يتم كتابة الأمر التالى :

GOTO 3

٢١ - ٣ - ٩ إستخدام الأمر (EDIT)

يستخدم الأمر (EDIT) عادة بعد وضع المؤشر على سجل معين ثم كتابة الأمر فى أبسط صورة له كالآتى :

EDIT

وفى هذه الحالة تظهر شاشة إدخال البيانات التى يتم عن طريقها تعديل البيانات المطلوبة.

٢١ - ٣ - ١٠ استخدام الأمر (APPEND)

يستخدم الأمر (APPEND) فى إضافة سجل بعد آخر سجل به بيانات فى الملف. ويكتب الأمر كالآتى :

```
APPEND
```

فتظهر شاشة إدخال خالية يتم عن طريقها إضافة السجل الجديد.

٢١ - ٣ - ١١ إنشاء واستخدام ملف الفهرس (Index File)

يتم إنشاء ملف الفهرس باستخدام الأمر (INDEX ON). فمثلا عندما يراد إنشاء ملف فهرس بناء على حقل الإسم (Name) مع تسمية هذا الملف (Name) يتم كتابة الأمر كالآتى :

```
INDEX ON Name TO Name
```

مع ملاحظة أن برنامج(DBase III +) يضيف للف الفهرس الإمتداد(NDX.).

ويتم فتح ملف الفهرس بطريقتين الطريقة الأولى عند فتح ملف قاعدة البيانات باستخدام الأمر التالى :

```
USE Myfile INDEX Name
```

حيث (Myfile) هو إسم ملف قاعدة البيانات
و (Name) هو إسم الملف الفهرسى الخاص بهذا الملف.

والطريقة الثانية باستخدام الأمر التالى

```
SET INDEX TO Name
```

٢١ - ٣ - ١٢ إنشاء واستخدام ملف الفرز (Sorting)

كما سبق الإيضاح فإن الفرز يؤدى إلى إنشاء ملف جديد مرتب بالترتيب المطلوب. ولتنفيذ هذه العملية باستخدام أوامر النقطة (Dot Commands) يتم كتابة الأمر التالى :

```
SORT TO Sname ON Name
```

حيث (Sname) هو إسم الملف الجديد المطلوب إنشاؤه.
و (Name) هو إسم الحقل الذى يتم الترتيب بناء عليه.

وفى هذه الحالة يتم إنشاء ملف جديد إسمه (Sname) مرتب حسب الترتيب الهجائى لحروف حقل الإسم مع ملاحظة أن الترتيب يكون تصاعديا (Ascending) من الأقل فالأكبر. فإذا أريد عكس هذا الترتيب يتم إضافة الحرف (D) فى نهاية الأمر كالآتى :

```
SORT TO Sname ON Name /D
```

ملاحظة

ماسبق ذكره فى هذا الباب ينطبق أيضا على كل برامج عائلة (DBase) مثل (DBase IV ، FoxBase ، FoxBase + ، FoxPro).

الفصل الثانى والعشرون

كتابة البرامج


Converted by Tiff Combine - (no stamps are applied by registered version)

من الإمكانيات المتقدمة لبرنامج (+ DBase III) وكذلك باقى برامج عائلة (DBase) مثل (DBase IV) و (+ FoxBase) و (FoxPro) أنها تستخدم كأداة برمجة قوية (Programming Tool). وهو ما يميزها عن كثير من اللغات الأخرى لسهولة كتابة البرامج بها واستخدام كثير من الدوال المبنية فيها (Built in Functions).

وكتابة البرامج تعتمد على تجميع أوامر النقطة (Dot Commands) فى ملف مع إضافة بعض أوامر التحكم التى تساعد على التحكم فى تسلسل تنفيذ الأوامر. ويتم تشغيل هذه الأوامر عن طريق تشغيل الملف الذى يحتوى عليها والذى يسمى ملف البرنامج (Program File).

ويمكن دراسة المثال التالى لتوضيح أهمية كتابة البرامج بواسطة (+DBase III).

مثال

نفرض أنه يوجد ملف قاعدة بيانات يسمى (Money.DBF) يحتوى على معلومات عن أشخاص مدينين ويراد معرفة بيانات الأشخاص الذين مازالوا مدينين وكذلك مجموع الديون المستحقة والحصول على تقرير بذلك. هذه العمليات يتم تنفيذها من خلال أوامر النقطة (Dot Commands) التالية :

```
USE MONEY INDEX NAME ↵
LIST FOR Owing ↵
CLEAR ↵
SUM Amount_due FOR Owing ↵
REPORT FORM MONEY FOR Owing TO PRINT ↵
USE ↵
```

وهذه الأوامر يمكن توضيحها كالآتى :

١ - الأمر الأول يؤدى إلى فتح الملف (Money.dbf) بالإضافة إلى فتح الفهرس (Name.ndx).
٢ - الأمر الثانى يؤدى إلى عرض بيانات الأشخاص المدينين.
٣ - الأمر الثالث يؤدى إلى مسح الشاشة.
٤ - الأمر الرابع يؤدى إلى تجميع المبالغ للأشخاص المدينين.

٥ - الأمر الخامس يؤدى إلى طباعة تقرير بيانات المدينين.
٦ - الأمر السادس يؤدى إلى إغلاق جميع الملفات المفتوحة.
٧ - علامة (↵) تعنى الضغط على مفتاح الإدخال بعد كل أمر.

ويمكن كتابة هذه الأوامر فى ملف وتسمية هذا الملف (Owing.prg). ويمكن تنفيذ نفس العمليات السابقة بكتابة أمر واحد أمام النقطة (Dot) وهو كالآتى :

DO Owing

والضغط على مفتاح الإدخال.

وهذا مثال بسيط ولكنه يوضح كيف يمكن توفير الوقت والجهد عن طريق كتابة البرامج من خلال (+ DBase III) وبرامج عائلة (DBase) الأخرى.

٢٢ - ١ أهمية كتابة البرامج

تسمح كتابة البرامج بواسطة برنامج (+ DBase III) أو أى برنامج من عائلة (DBase) بتسهيل تعامل المستخدم مع قواعد البيانات. وذلك عن طريق عرض القوائم الواضحة التى يستطيع من خلالها تنفيذ عمليات برامج إدارة قواعد البيانات مثل إدخال سجلات جديدة أو تعديل بيانات السجلات أو تحديث البيانات المخزنة أو عرض البيانات أو طباعتها وهكذا. كما تسمح كتابة البرامج أيضا بوضع وسائل التأمين للبيانات بحيث لايتعامل مع هذه البيانات إلا الأشخاص المكلفون بذلك. كما أنها تتيح للمستخدم استخدام كل خواص برامج ادارة قواعد البيانات دون الحاجة لوجود خبرة سابقة بهذه البرامج أى أنها تمثل حلقة الإتصال بين المستخدم وبين قاعدة البيانات.

٢٢ - ٢ إنشاء ملف البرنامج (Program File)

لإنشاء ملف البرنامج يستخدم الأمر التالى :

MODIFY COMMAND

ثم إسم الملف المطلوب إنشاؤه.

فإذا كان هذا الملف موجودا على القرص تظهر قائمة الأوامر الخاصة به على الشاشة. وإذا لم يكن موجودا يتم فتح ملف جديد مع ملاحظة عرض قائمة مساعدة (Help Menu) أعلى الشاشة لتوضيح مفاتيح التصحيح المطلوب استخدامها أثناء كتابة البرنامج. ويمكن إلغاء هذه القائمة بالضغط على مفتاح (F1) إذا أريد الإستفادة بالشاشة كلها فى كتابة البرنامج.

وعند الإنتهاء من كتابة البرنامج يتم تخزينه باستخدام مفتاحى (Ctrl-End) أو مفتاحى (Ctrl-W). وعند عدم الرغبة فى تخزين الملف أو التعديلات التى تم إدخالها عليه يتم الضغط على مفتاح (Esc) فيظهر السؤال التالى على الشاشة :

Abort editing ? (Y/N)

وعند كتابة (Y) يعود البرنامج إلى مشيرة النقطة (Dot Prompt).

ملاحظة

ما سبق ذكره فى هذا الباب ينطبق أيضا على كل برامج عائلة (DBase) مثل (DBase IV) ، (FoxBase +) ، (FoxPro).



الفصل الثالث والعشرون

خصائص كتابة البرامج



٢٣ - ١ ماهو البرنامج

البرنامج هو مجموعة من الإرشادات والأوامر التى توجه الحاسب إلى تنفيذ مهام أو وظائف معينة بتسلسل محدد. والبرنامج بصفة عامة لايختص بالحاسب فقط ولكن أى مجموعة من الأوامر والإرشادات المرتبة بتسلسل معين هى برنامج. فمثلا الوصفة الخاصة بالطهى والتى تشمل خطوات محددة ومتسلسلة بحيث لايمكن مثلا طهى البطاطس قبل تقشيرها وتقطيعها هذه الوصفة تعتبر برنامج.

والفرق بين البرنامج الذى يعطى للإنسان والبرنامج الذى يعطى للحاسب هو أن الأول يمكن للإنسان تعديل بعض الأوامر فيه حسب إختياره بين البدائل المختلفة. أما بالنسبة للحاسب فإنه يلتزم بالأوامر الموجودة ولاينفذ إلا الأوامر التى تخضع لقواعد معينة سبق تحديدها.

لذلك فإن كتابة البرامج للحاسب تتطلب كتابة جميع الأوامر والارشادات بتسلسل دقيق حتى تؤدى الوظيفة المطلوبة.

٢٣ - ٢ لغة كتابة البرامج

لغة كتابة البرامج بواسطة (DBase III +) أو برامج عائلة (DBase) الأخرى مثل (DBase IV) ، (FoxBase +) ، (FoxPro) هى لغة سهلة وواضحة لأنها تحتوى على الكلمات الإنجليزية الواضحة مثل (IF) ، (DO) ، ... الخ. ولكن الحاسب فى الواقع لايفهم هذه الكلمات ، لذلك فإن المترجم (Interpreter) الخاص بالبرنامج يقوم بقراءة كل أمر يتم إدخاله وترجمته إلى لغة الآلة (Machine Language) التى يفهمها الحاسب. فمثلا عند إدخال الأمر (USE) فإن الحاسب يقوم بمقارنة كل حرف فى الأمر (U-S-E) بقائمة الأوامر (Instruction Set) المخزنة فى الحاسب وهذه الخطوة تسمى (Parsing) أى تقسيم الأمر. وعندما يجد المترجم (Interpreter) تطابقا بين هذا الأمر الذى تم إدخاله وبين أحد الأوامر الموجودة فى قائمة الأوامر فإنه ينفذ التعليمات الخاصة بهذا الأمر. وعندما لايجد الأمر مطابقا فإنه يعرض رسالة خطأ (Error message). وأخطاء كتابة البرنامج يمكن أن تحدث لأحد الأسباب الآتية :

١ - أخطاء فى هجاء الأمر.
٢ - عدم فصل الأمر عن المعاملات (Parameters) بمسافة خالية (Space) على

الأقل.

٣ - أخطاء القواعد (Syntax Errors) وتعنى عدم تطبيق القواعد الخاصة بهذا الأمر تطبيقا سليما.

٤ - أخطاء فى التسلسل المنطقى للأوامر.

٢٣ - ٣ كتابة وتصحيح البرنامج

يحتوى برنامج (+ DBase III) على برنامج معالجة كلمات مبنى داخل البرنامج (Built in) يستخدم فى كتابة وتصحيح ملفات البرامج. ولتشغيل برنامج معالجة الكلمات يستخدم الأمر (MODIFY COMMAND) ثم كتابة إسم الملف المطلوب كتابته. والبرنامج يضيف الإمتداد (prg.) آليا إلى إسم الملف المطلوب إنشاؤه. فمثلا عند كتابة برنامج يراد تسميته (Test.prg) يتم إدخال الأمر الآتى بعد مشيرة النقطة (Dot prompt).

MODIFY COMMAND Test

والضغط على مفتاح الإدخال.

وعندما يراد تعديل ملف برنامج موجود يتم استخدام نفس الأمر (MODIFY COMMAND) وبعده إسم الملف. وعند فتح ملف البرنامج تظهر قائمة المساعدة (Help menu) أعلى الشاشة لتوجه مخطط البرامج إلى المفاتيح المستخدمة فى تحريك المؤشر (Cursor) وأداء عمليات التصحيح المختلفة. ولإخفاء هذه القائمة يستخدم المفتاح (F1). كما يمكن إعادتها مرة ثانية باستخدام نفس المفتاح. انظر الشكل (٢٣ - ١).

ويمكن استخدام أى برنامج معالجة كلمات (Word Processor) أو مصحح نصوص (Text Editor) فى كتابة برامج (+ DBase III) على أن يكون من البرامج التى تسمح بإنشاء ملفات آسكى (ASCII). وعند استخدام أى برنامج معالجة كلمات غير البرنامج المستخدم يجب التأكد أن الملف لايحتوى على أى شفرة تشكيل (Formating Code) مثل الحروف البارزة (Boldface) أو الخطوط السفلية (Underline) أو أى تشكيل آخر للصفحة.

```
CURSOR  <-- -->  | INSERT     | DELETE     | Up a field:    ↑
 Char:    ← →    | Char:  Ins | Char:  Del | Down a field:  ↓
 Word: Home End  | Field: ^N  | Word:  ^Y  | Exit/Save:    ^End
 Pan:   ^← ^→    | Help:  F1  | Field: ^U  | Abort:         Esc
```

شكل (٢٣ - ١)

ومن البرامج التى توفر كتابة نصوص بهذه الطريقة برنامج (Framework II) وبرنامج (Wordstar) الذى يسمح بتكوين ملفات غير وثائقية (Nondocument). وعند كتابة البرامج بواسطة أى برنامج معالجة كلمات يجب إضافة الإمتداد (.prg) إلى إسم البرنامج.

ملاحظة

إذا كانت الذاكرة المؤقتة (RAM) فى الحاسب كافية يمكن كتابة إسم برنامج معالجة الكلمات المراد استخدامه داخل ملف المواصفات (Config.sys) بحيث يصبح هو المصحح المبدئى (Default).

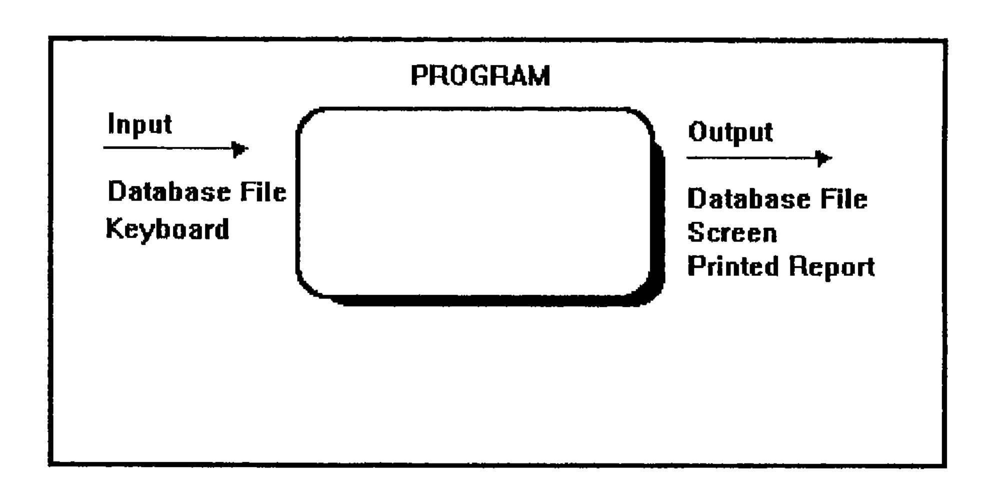
ويفضل استخدام برامج معالجة الكلمات عندما يكون ملف البرنامج المطلوب كتابته كبيرا. حيث أن المصحح الموجود فى برنامج (+ DBase III) يكتب حتى ٥٠٠٠ حرف فقط أى ما يقرب من ٢٠٠ سطر.

٢٣ - ٤ تشغيل البرنامج

عندما يراد تشغيل البرنامج يتم كتابة الأمر (DO) ثم إسم ملف البرنامج الذى سبق إنشاؤه فيقوم برنامج (+ DBase III) بفتح ملف البرنامج وقراءة كل سطر فيه بدءا من أول سطر مع تنفيذ كل أمر بالتسلسل. وعندما يصل إلى آخر أمر وينفذه يعود إلى مشيرة النقطة (Dot prompt) مرة ثانية.

٢٣ - ٥ المدخلات والمخرجات (Input and Output)

أى برنامج تكون له مدخلات ومخرجات وهى التى تمثل مصدر المعلومات(Source) ومكان وصول هذه المعلومات (Destination) على الترتيب. فمثلا المدخلات يمكن أن تأتى من حقول قاعدة البيانات المستخدمة ويمكن أن يدخلها المستخدم عن طريق لوحة المفاتيح (Keyboard) أثناء تشغيل البرنامج. أما المخرجات فقد تكون قوائم بيانات السجلات على الشاشة أو التقارير المطبوعة فى الطابعة. انظر الشكل (٢٣ - ٢).



شكل (٢٣ - ٢) المدخلات والمخرجات

والبرنامج الذى يتم كتابته يجب أن يوفر التحكم فى المدخلات والمخرجات بأقل تدخل من المستخدم. حيث تنحصر وظيفة المستخدم فى إدخال البيانات والحصول على المعلومات المطلوبة بعرضها على الشاشة أو طباعتها على الطابعة.

ويتم التحكم فى المدخلات والمخرجات عن طريق الشاشات المجهزة (Customized Screens). وبالنسبة للمدخلات تعمل هذه الشاشات كمحادثة مباشرة (Dialog) بين المستخدم والحاسب مع الرسائل التوضيحية التى تساعد على توجيه المستخدم أو تحذيره عند حدوث أى خطأ فى إدخال البيانات. كما يمكن تصميم شاشات إدخال البيانات لتماثل النماذج المستخدمة فى بعض النظم حتى يتمكن المستخدم من إدخال البيانات من النماذج الموجودة عنده بسهولة. وهذا يؤدى إلى سهولة تشغيل البرنامج بواسطة أى شخص دون الحاجة إلى أشخاص مؤهلين ذوى خبرة ببرامج إدارة قواعد

البيانات. انظر الشكل (٢٣ - ٣)

```
Set Up              Modify              Options              Exit

  _ CADETS INFORMATION

CADET NO  99999     RELATIUES  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
HOBBIES   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX   DATE_SEC  99999
NO_BROTHER  99999          FATHER JOB               XXXXXXXXXXXXXX

CREATE SCREEN  ||<A:>||A:CADETS.SCR          ||Pg 01 Row 03 Col 23||Ins
     Enter text.  Drag field or box under cursor with ←┘.  F10 for mo
                  Screen field definition blackboard
```

شكل (٢٣ - ٣) شاشة الادخال

وبالنسبة للمخرجات يمكن أيضا استخدام الشاشات المجهزة (Customized Screens) فى عرض البيانات التى يقوم المستخدم بإدخالها واعطائه الفرصة للتأكد من صحتها (Validation) قبل تخزينها وكذلك إعطائه الفرصة لاسترجاع أى بيانات على الشاشة أو طباعتها على الطابعة.

وعن طريق البرنامج أيضا يمكن تخزين البيانات التى يقوم المستخدم بإدخالها فى أماكن مؤقتة فى الذاكرة تسمى متغيرات الذاكرة (Memory Variables). ولايتم نقل هذه البيانات إلى الحقول الخاصة بها فى قاعدة البيانات إلا بعد أن يتأكد المستخدم من صحة البيانات التى قام بإدخالها. وهذه الطريقة تفيد فى تحقيق البيانات(Validation) والتأكد من صحتها. حيث أن البيانات غير الصحيحة تؤدى دائما إلى مخرجات غير صحيحة. وهو مايعرف فى عالم الحاسب (Garbage in Garbage out) ويختصر (GIGO). وهو يعنى أن جودة المخرجات ترتبط ارتباطا كبيرا بجودة المدخلات.

وعن طريق البرنامج أيضا يمكن تحقيق تكامل قاعدة البيانات (Database Integrity) ووحدة البيانات. فإذا كان هناك عدة مستخدمين يقومون بإدخال البيانات يتم التأكد من دخول البيانات السليمة بواسطة الأشخاص المسموح لهم بإدخال هذه البيانات. وتفيد متغيرات الذاكرة (Memory Variables) أيضا فى تحقيق ذلك وهذه المتغيرات سيتم شرحها بالتفصيل فيما بعد.

وعند كتابة البرنامج يجب دراسة كل الأخطاء المتوقعة من المستخدم وكتابة الأوامر التي تؤدى إلى عدم توقف البرنامج بالإضافة إلى تنبيه المستخدم إلى الخطأ وطريقة إصلاحه. فمثلا عندما يكون هناك سؤال ينتهى بالإختيار بين (Yes) أو (No) يجب التأكد أن البرنامج يعرف ماذا يفعل عندما يدخل المستخدم أى حرف آخر غير (N) أو (Y) عن طريق الخطأ. وهذا سيتم إيضاحه فيما بعد بالتفصيل.

٢٣ - ٦ التحكم فى البرنامج

كما سبق الإيضاح فإن المترجم (Interpreter) يقوم بقراءة أوامر البرنامج وترجمتها بالتسلسل من البداية إلى آخر أوامر البرنامج. ثم يقوم الحاسب بتنفيذ كل أمر فور ترجمته بنفس التسلسل.

ولكن فى بعض الأحيان يراد تنفيذ بعض الأوامر عدة مرات أو يراد الرجوع إلى بعض الأوامر التى سبق تنفيذها أو يراد تخطى بعض الأوامر والذهاب إلى أوامر فى مكان آخر من البرنامج. وفى جميع هذه الأحوال يراد تغيير تسلسل تنفيذ أوامر البرنامج.

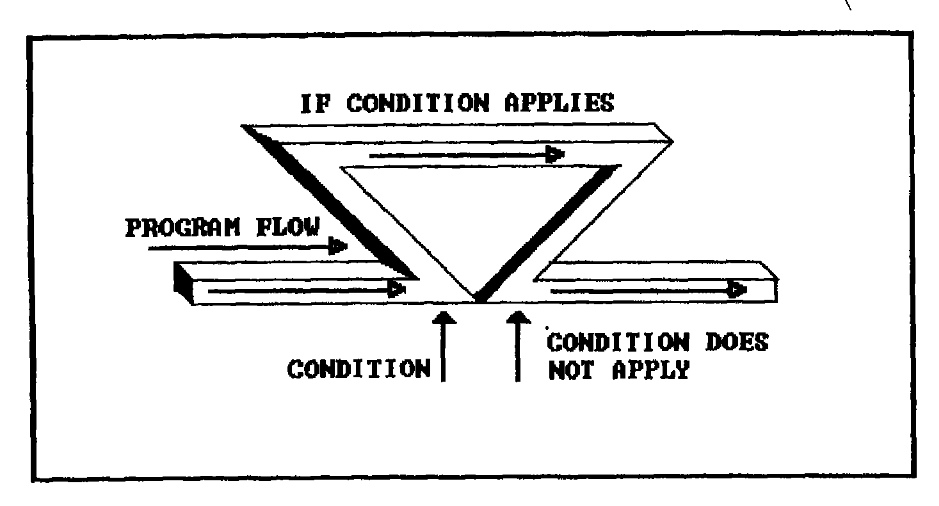
ولتنفيذ ذلك تستخدم أوامر التحكم التى يؤدى بعضها إلى تفرع البرنامج إلى مكان آخر تفرعا غير مشروط وبعضها يؤدى إلى تفرع البرنامج تفرعا مشروطا يتوقف على تحقيق شرط معين أو عدم تحقيقه وبعضها يؤدى إلى تكرار تنفيذ مجموعة من الأوامر بناء على شرط معين. وهذه الأوامر تعطى مخطط البرامج قدرة ومرونة عالية.

٢٣ - ٦ - ١ التفرع المشروط

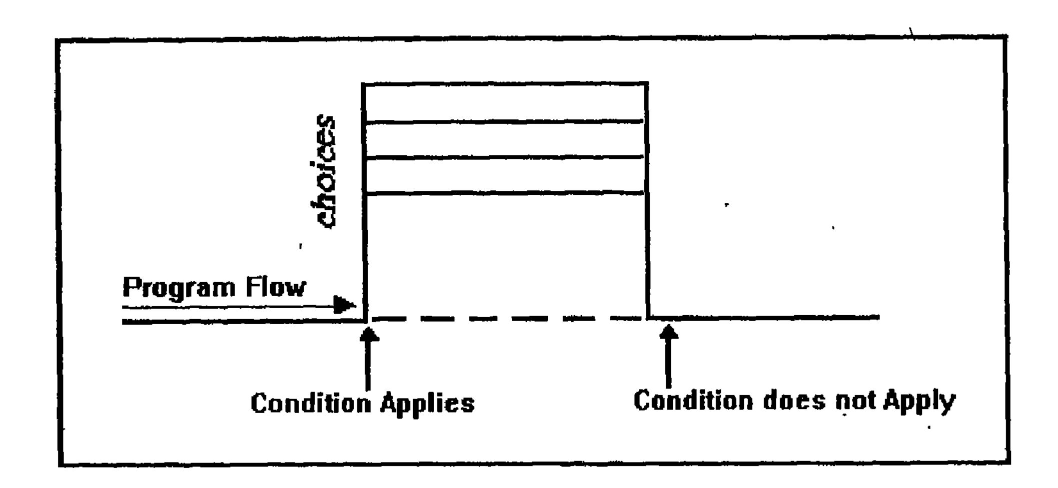
فى بعض الأحيان يراد تنفيذ بعض العمليات التى ترتبط بشرط معين (Condition). وذلك عندما يراد مثلا إدخال زيادة فى مرتب الموظفين الذين تزيد مدة خدمتهم عن ١٠ سنوات. ولتنفيذ ذلك يتيح البرنامج طريقتين لتنفيذ هذه العملية الأولى باستخدام (IF-ENDIF) وذلك عندما يكون هناك إحتمالان فقط للشرط. انظر الشكل (٢٣ - ٤)

أما إذا كانت هناك عدة إحتمالات فيستخدم الأمر (DO CASE - ENDCASE). وذلك عندما يراد مثلا عرض قائمة إختيارات على الشاشة والتفرع إلى البرنامج الذى ينفذ الإختيار المطلوب للمستخدم. انظر

الشكل (٢٣ - ٥)



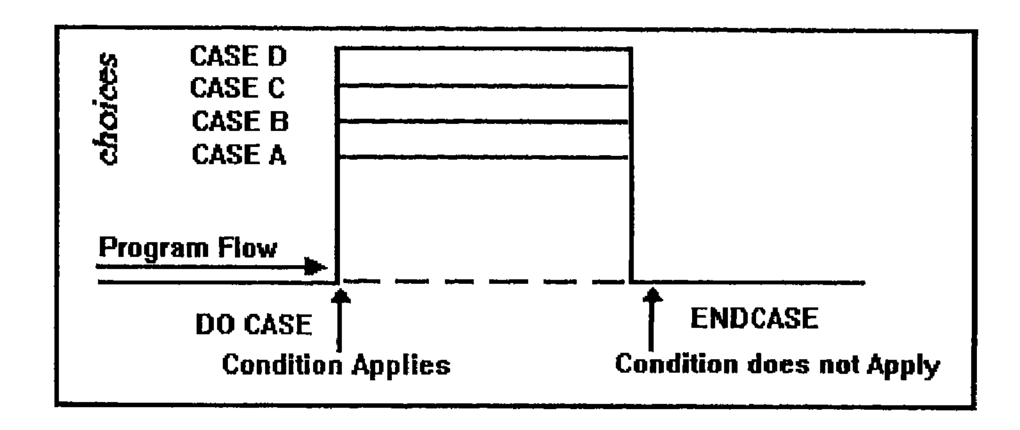
شكل (٢٣ - ٤)



شكل (٢٣ - ٥)

والقوائم تستخدم فى معظم برامج إدارة قواعد البيانات وتبدأ عادة بقائمة رئيسية (Main Menu) تتفرع إلى قوائم أخرى فرعية (Submenus) تظهر عند اختيار المستخدم لأحد اختيارات القائمة الرئيسية. ويستخدم الأمر (DO CASE-ENDCASE) فى التفرع من البرنامج بناء على الشرط الذى يلى الأمر (DO CASE). حيث يتم تنفيذ كل مجموعة من الأوامر حسب القيمة الموجودة

بعد كلمة (CASE). أنظر الشكل (٢٣ - ٦)



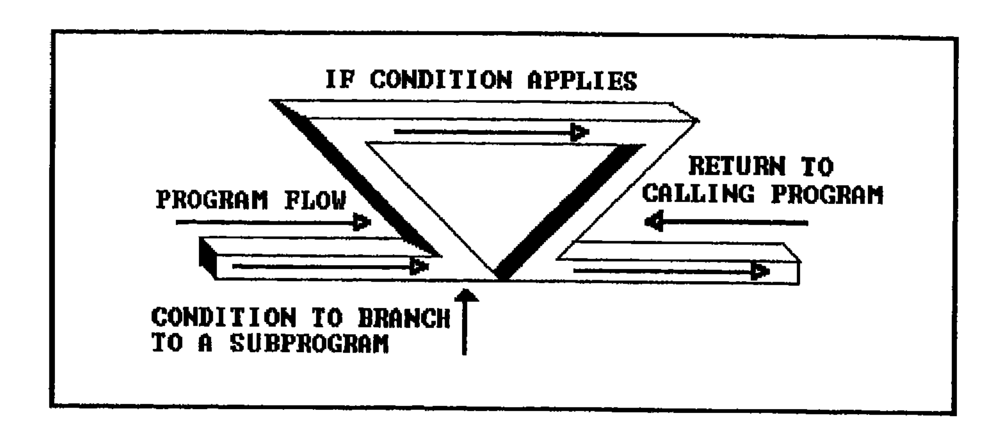
شكل (٢٣ - ٦)

٢٣ - ٦ - ٢ التفرع الى برنامج فرعي

كما سبق الإيضاح فى المثال الخاص بوصفة الطهى (Cooking Recipe) ، تكون هناك مجموعة من الإرشادات والتعليمات لطهى نوع معين من الأطعمة وهذه الإرشادات تكون فى الواقع برنامجا مثل برامج الحاسب. وعندما يراد مثلا عمل فلفل مقلى ضمن الوجبة فإن الطاهى يلجأ إلى وصفة أخرى خاصة بقلى الفلفل وينفذها ثم يعود إلى الوصفة الأولى ويكمل تنفيذ التعليمات الموجودة بها. وهذه الوصفة الأخرى الخاصة بقلى الفلفل تمثل برنامجا فرعيا متفرعا من البرنامج الأول.

وعند كتابة البرنامج يمكن التفرع إلى برامج أخرى لتنفيذ العمليات التى يحتاجها المستخدم. فمثلا عندما تكون هناك قائمة اختيارات يختار منها المستخدم اختيارا معينا فإن البرنامج الرئيسى ينتقل إلى برنامج آخر ينفذ هذا الإختيار.

وعند الإنتهاء من تنفيذ البرنامج الفرعى يعود البرنامج مرة أخرى إلى نفس المكان الذى انتقل منه. وهذه الخاصية تعطى قوة ومرونة كبيرة للبرنامج لأنها تسمح بتقسيم أى برنامج كبير إلى عدة برامج صغيرة (Modules) مما يسهل اختبار كل برنامج صغير وتصحيحه مستقلا عن باقى البرامج. انظر الشكل (٢٣ - ٧)



شكل (٢٣ - ٧) التفرع إلى برنامج فرعى

ولتنفيذ هذه العملية يستخدم الأمر (DO) وبعده إسم البرنامج المطلوب تنفيذه. ويكتب هذا الأمر داخل البرنامج الرئيسى الذى يسمى فى هذه الحالة برنامج الاستدعاء (Calling Program). وعندما يجد البرنامج الأمر (DO) فإنه يذهب إلى البرنامج الفرعى (Module) وينفذه ثم يعود إلى الأمر التالى للأمر (DO) مباشرة ويكمل تنفيذ البرنامج. وسوف يتم شرح هذا الأمر بالتفصيل فيما بعد.

٢٣ - ٦ - ٣ الحلقة التكرارية (LOOP)

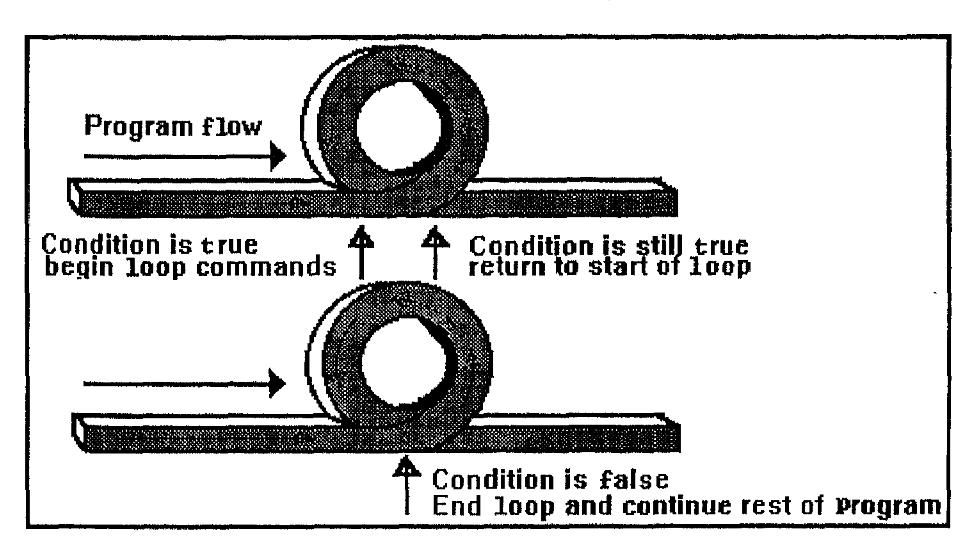
فى بعض الأحيان يراد تنفيذ مجموعة من الأوامر عددا من المرات يتوقف علي تحقق شرط معين. وهذه المجموعة من الأوامر تسمى الحلقة التكرارية (Loop). وهى تبدأ بالأمر (DO WHILE) وتنتهى بالأمر (ENDDO). وهذان الأمران يمثلان حدود الحلقة التكرارية.

فمثلا عندما يراد عرض بيانات المتزوجين فقط فى قاعدة البيانات الخاصة بالموظفين يستخدم الأمر التالى :

DO WHILE Married

حيث (Married) هو إسم الحقل المنطقى الذى يحدد إذا كان الموظف متزوجا أو غير متزوج. وعند كتابة الأمر بهذا الشكل فإن البرنامج ينفذ الأوامر التالية له طالما

كان الموظف متزوجا لأن الحقل فى هذه الحالة يكون صحيحا (True). أما الموظف غير المتزوج فإن الحقل الخاص به يكون غير صحيح (False) وبالتالى لايتم عرض بيانات الموظفين الغير متزوجين. انظر الشكل (٢٣ - ٨)



شكل (٢٣ - ٨)

ويلاحظ من هذا الشكل أن تدفق البرنامج يصل إلى شرط معين مطلوب تحقيقه فإذا تحقق الشرط يتم تنفيذ أوامر الحلقة وعندما يصبح الشرط غير صحيح يتوقف تنفيذ الحلقة ويستمر تدفق البرنامج فى إتجاهه المعتاد.

٢٣ - ٧ الإعداد للبرنامج

قبل البدء فى كتابة البرنامج يجب أولا دراسة مخرجات البرنامج المطلوبة مثل التقارير المطبوعة أو المعروضة على الشاشة. وهذه المرحلة تتطلب إشتراك المستخدم مع مصمم البرنامج لتحديد المخرجات المطلوبة وكذلك لتحديد الآتى :

١ - الهدف من البرنامج.
٢ - كيفية تحقيق البرنامج لهذا الهدف.
٣ - نوع وشكل شاشات المساعدة (Help Screens).
٤ - تحديد إمكانية تحسين البرنامج وتطويره.
٥ - تحديد مصدر مدخلات البرنامج وإذا كانت عن طريق لوحة المفاتيح أو من ملفات معينة.

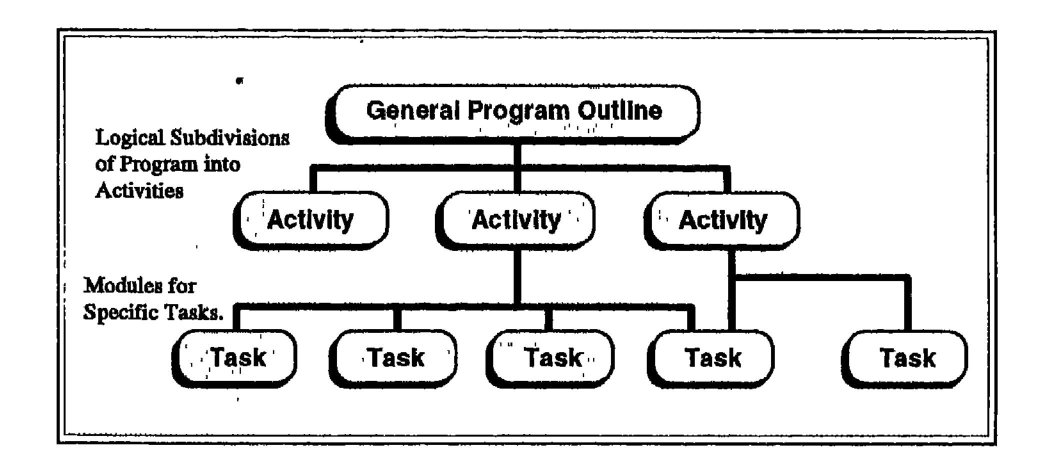
٢٣ - ٨ التصميم من أعلى إلى أسفل (Top-Down Design)

يبدأ تصميم البرنامج بتحديد الخطوط العامة (General Outline) التى توضح خطوات تصميم البرنامج بالتسلسل المنطقى الذى يؤدى إلى تحقيق الهدف النهائى للبرنامج. فمثلا عندما يراد كتابة برنامج يقوم بعرض بيانات سجل معين فى قاعدة بيانات معينة يتم كتابة الخطوات التالية لتحديد الخطوط العامة للبرنامج :

١ - تجهيز محيط التشغيل (Working Environment) وذلك بتحديد الفهرس الفرعى المستخدم وفتح الملفات المستخدمة ومسح الشاشة و ... الخ.
٢ - تحديد السجل المطلوب عرضه.
٣ - تحديد الحقول المطلوب عرضها فى السجل.
٤ - عرض الحقول المطلوبة فى السجل.
٥ - سؤال المستخدم إذا كان يريد عرض سجل آخر أم لا.
٦ - إذا كان المستخدم يريد عرض سجل آخر يتم إعادة الخطوات ٢ ، ٣، ٤.
٧ - إذا كان المستخدم لايريد عرض سجل آخر يتم إغلاق جميع الملفات والعودة إلى مشيرة النقطة (Dot Prompt).

وبعد تحديد الخطوط العامة (General Outline) يتم تحديد الوظائف الرئيسية للبرنامج والتى تحقق هذه الخطوات. ثم يتم فصل هذه الوظائف بالتدريج إلى وظائف أصغر فأصغر ثم كتابة البرامج التى تؤدى إلى تنفيذ هذه الوظائف الصغيرة. هذه البرامج الصغيرة التى تسمى (Modules) يؤدى كل منها وظيفة محددة ومستقلة من وظائف البرنامج الرئيسى. كما أن هذا البرنامج الصغير (Module) يمكن أن يتكرر استخدامه داخل البرنامج الرئيسى عندما يتطلب البرنامج تنفيذ الوظيفة الخاصة عدة مرات. كما يمكن استخدام نفس هذا البرنامج الفرعى مع برامج رئيسية أخرى. وتسمى هذه الطريقة فى كتابة البرامج البرمجة التركيبية (Structural Programming).

وهذه الطريقة تمتاز عن الطريقة التقليدية بسهولة كتابة كل برنامج واختباره منفصلا. كما أن هذه البرامج الصغيرة يمكن استخدامها فى برامج أخرى متعددة لتحقيق وظائف محددة فى هذه البرامج. كما يمكن تجميع مجموعة كبيرة من هذه البرامج الصغيرة فى مكتبة برامج (Library) يمكن استخدامها عند الحاجة. انظر الشكل (٢٣ - ٩)



شكل (٢٣ - ٩)

ومن فوائد هذه الطريقة التركيبية أيضا أنها تساعد على تطوير البرنامج فيما بعد أو توسيعه (Expansion) بسهولة. حيث يكفى فى هذه الحالة إضافة برامج فرعية جديدة تحقق وظائف إضافية للبرنامج الرئيسى.

٢٣ - ٩ كتابة الملاحظات فى البرنامج (Comments)

من المهم جدا وضع ملاحظات داخل البرنامج لتوضيح خطوات البرنامج وذلك حتى يسهل بعد ذلك تصحيحه أو تطويره. وسطر الملاحظات يبدأ بكلمة (NOTE) أو الحرف (*). وعندما يقوم البرنامج المترجم (Interpreter) بترجمة شفرة البرنامج فإنه يمر على سطور الملاحظات دون ترجمتها ولكن هذه الملاحظات تظهر عند عرض سطور البرنامج على الشاشة عند تصحيحه. ومن الملاحظات السطور التالية مثلا :

```
*      This is a comment line
Note this line is not a command line.
```

ويمكن إضافة الملاحظات فى نفس سطر الأمر لتوضيح وظيفة هذا الأمر وذلك بكتابة الحرفين (&&) قبل الملاحظات كالآتى :

```
DO Menu && Run the Menu Program
```

ملاحظة

يجب فصل الحرفين (&&) عن الأمر الموجود فى السطر بمسافة واحدة (Space) على الأقل.

وهذه الملاحظات تسهل تتبع أوامر البرنامج وتسلسله المنطقى واكتشاف أى أخطاء. كما أنها تساعد أى شخص غير الشخص الذى قام بكتابة البرنامج على تعديل البرنامج أو تطويره أو إكتشاف أى أخطاء أو مشاكل به.



الفصل الرابع والعشرون

تركيب البرنامج

(Program Structure)



يمكن تقسيم البرنامج المكتوب إلى أربعة أقسام رئيسية وهى المقدمة وأوامر التجهيز (Set Up) وأوامر البرنامج وأوامر الخروج.

٢٤ - ١ المقدمة

وهى إختيارية يمكن كتابتها أو عدم كتابتها ويتم فيها كتابة إسم مصمم البرنامج وتاريخ تصميمه وأى معلومات أخرى مطلوب إضافتها. وتستخدم لكتابة سطورها كلمة (NOTE) أو الحرف (*) فى بداية كل سطر.

٢٤ - ٢ أوامر التجهيز (Setup)

وهى الأوامر التى تؤدى إلى تجهيز محيط التشغيل (Working Invironment) لاستقبال أوامر البرنامج. وبعض هذه الأوامر يبدأ بالأمر (SET) مثل الأوامر التالية :

```
SET TALK OFF
SET DEFAULT TO
```

وسوف يتم دراستها بالتفصيل فيما بعد.

وهناك أوامر أخرى تستخدم فى فتح ملفات قاعدة البيانات وملفات الفهرس (Index) والبحث (Query) و ... الخ. وهناك أوامر أخرى تستخدم فى إنشاء متغيرات الذاكرة (Memory Variables) التى تستخدم فى التخزين المؤقت للمدخلات والمخرجات وسوف يتم دراستها فيما بعد.

وأوامر التجهيز عادة يكتب معظمها فى البرنامج الرئيسى وتصبح فى هذه الحالة مؤثرة فى البرنامج الرئيسى والبرامج الفرعية فى نفس الوقت. كما أن هناك بعض أوامر التجهيز التى تكتب فى برامج فرعية معينة وفى هذه الحالة لاتؤثر إلا فى هذه البرامج الفرعية فقط.

٢٤ - ٣ أوامر البرنامج

وهى الأوامر الرئيسية فى البرنامج التى تؤدى المهام الرئيسية مثل استقبال المدخلات

(Input) من المستخدم وعرض البيانات وتعديل البيانات وإنشاء المخرجات (Output). وهى تشمل أوامر التحكم والتكرار (Looping) والتفرع إلى برامج فرعية أخرى لتنفيذ المهام المطلوبة من البرنامج.

٢٤ - ٤ أوامـــر الخـروج

وهى الأوامر التى تؤدى إلى الخروج من البرنامج والعودة إلى مشيرة النقطة (Dot Prompt). وهى تشمل أوامر إغلاق الملفات المفتوحة (Closing Commands) للمحافظة على وحدة وتكامل قاعدة البيانات (Database Integrity) ومنها الأوامر التالية :

```
CLOSE DATABASES
CLEAR ALL
USE
```

ثم تأتى أوامر الخروج مثل :

```
RETURN
QUIT
```

والأمر (RETURN) إذا كتب فى البرنامج الرئيسى فإنه يؤدى إلى العودة إلى مشيرة النقطة (Dot Prompt) أما إذا كتب فى برنامج فرعى فإنه يؤدى إلى العودة إلى البرنامج الذى قام باستدعائه (Calling Program). أما الأمر (Quit) فإنه يؤدى إلى الخروج من برنامج (+ DBase III) والعودة إلى نظام التشغيل.

٢٤ - ٥ إستخدام الأمر (DO)

يستخدم الأمر (DO) كما سبق الإيضاح فى التفرع من البرنامج الرئيسى إلى برنامج فرعى. كما يمكن أيضا التفرع من برنامج فرعى إلى برنامج فرعى آخر وهذا التفرع يكون غير مشروط (Uncoditional). فعندما يقابل مترجم البرنامج (Interpreter) الأمر (DO) فإنه ينتقل إلى البرنامج الفرعى المحدد بالإسم الموجود بعد الأمر (DO) ويستمر فى تنفيذ أوامر البرنامج الفرعى حتى يصل إلى الأمر (RETURN) الذى يؤدى إلى الرجوع إلى البرنامج القائم بالإستدعاء (Calling Program).

فمثلا يمكن أن يحتوى برنامج على الأوامر التالية :

```
CLEAR            && clears the screen
USE CADETS         && opens cadets.dbf file
DO C_EDIT         && Branches to C_EDIT.prg
DO C_DEL          && Branches to C_DEL.prg
```

حيث (C_EDIT) هو برنامج فرعى يؤدى إلى تصحيح البيانات فى ملف قاعدة البيانات و (C_DEL) هو برنامج فرعى آخر يؤدى إلى مسح السجلات المطلوب إلغاؤها من ملف قاعدة البيانات. ويلاحظ هنا كتابة إسم البرنامج الفرعى بدون الإمتداد (Extension) الذى يضيفه البرنامج آليا ويكون دائما (.prg).

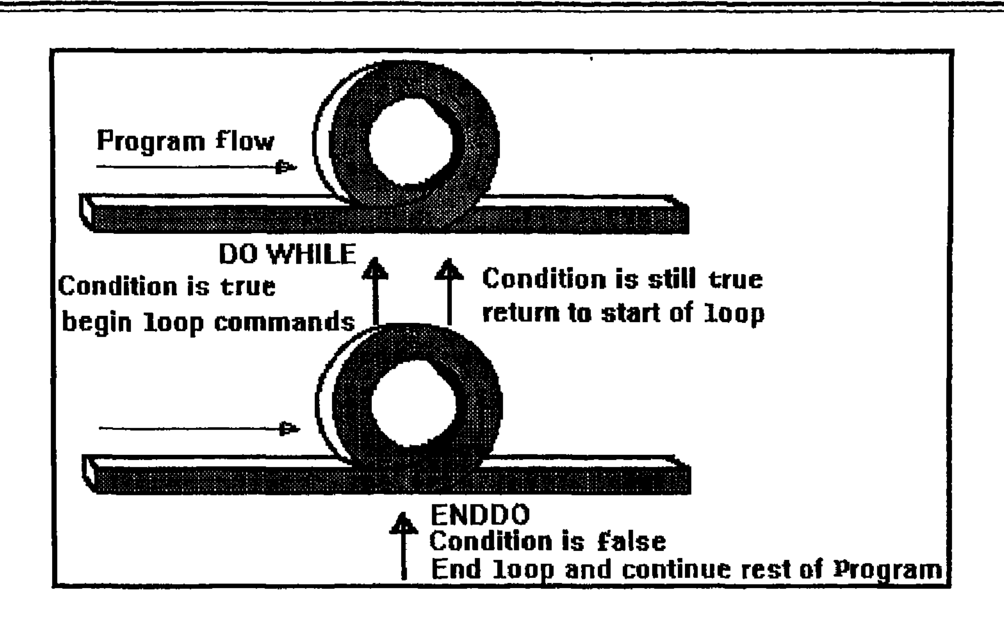
وعندما يصل البرنامج إلى الأمر (DO C_EDIT) مثلا فإنه ينتقل إلى برنامج (C_EDIT) وينفذ أوامره ثم يعود إلى البرنامج مرة ثانية عندما يقابل الأمر (RETURN) الذى يكون آخر أمر فى البرنامج الفرعى.

ملاحظة

يمكن استخدام الأمر (RETURN TO MASTER) للرجوع مباشرة إلى البرنامج الرئيسى بصرف النظر عن عدد البرامج الفرعية التى تم التفرع اليها. وهذا يؤدى إلى سرعة الرجوع إلى البرنامج الرئيسى دون المرور على عدة برامج فرعية.

٢٤ - ٦ إستخدام الأمر (DO WHILE)

يستخدم هذا الأمر كما سبق الإيضاح فى تكرار تنفيذ مجموعة من الأوامر عددا من المرات يتوقف على تحقق شرط معين يكتب بعد الأمر حيث يتم اختبار الشرط فى بداية الحلقة التكرارية (Loop) فإذا كان صحيحا (TRUE) يتم تنفيذ جميع الأوامر التالية للأمر (DO WHILE) ثم يتوقف عند الأمر (ENDDO) الذى يؤدى إلى العودة إلى بداية الحلقة التكرارية. ثم يتم اختبار الشرط مرة ثانية فإذا كان صحيحا يتم تنفيذ أوامر الحلقة وهكذا. وعندما يصبح الشرط غير صحيح (FALSE) يتوقف تنفيذ الحلقة وينتقل البرنامج إلى الأمر التالى للأمر (ENDDO). أنظر الشكل (٢٤ - ١)



شكل (٢٤ - ١)

فمثلا يمكن أن يحتوى برنامج على الأوامر التالية :

```
DO WHILE Age < = 40
    --------
    --------
    --------        Commands
    --------
    --------
ENDDO
```

وفى هذه الحالة يتم تنفيذ أوامر الحلقة طالما كان العمر أقل من أو يساوى ٤٠ سنة ويستمر تنفيذ أوامر الحلقة التكرارية حتى يصبح العمر أكبر من ٤٠ سنة. وفى هذه الحالة ينتقل البرنامج إلى الأوامر التالية للأمر (ENDDO).

كما يمكن أن يحتوى برنامج آخر على الحلقة التكرارية التالية :

```
DO WHILE .NOT. EOF()
```

```
    DISPLAY Name, Address, Age
    SKIP
ENDDO
```

ويلاحظ فى هذه الحالة استخدام المعامل المنطقى (NOT) وكذلك استخدام الدالة (EOF()) وهى تعنى (End of file) أى نهاية ملف قاعدة البيانات المفتوح. وتؤدى الحلقة التكرارية (Loop) إلى عرض حقول الإسم والعنوان والعمر لجميع السجلات حتى يصل المؤشر إلى آخر سجل فى الملف.

وتبدأ الحلقة باختبار وصول المؤشر إلى نهاية الملف فإذا لم يصل إلى نهاية الملف يصبح الشرط صحيحا وبالتالى يتم تنفيذ أوامر الحلقة ويتم عرض حقول الإسم والعنوان والعمر لهذا السجل. ثم يؤدى الأمر (SKIP) إلى الإنتقال إلى السجل التالى. كما يؤدى الأمر (ENDDO) إلى العودة إلى أول أمر فى الحلقة التكرارية ويتم اختبار الشرط مرة ثانية وهكذا يتكرر تنفيذ الحلقة التكرارية حتى يصل البرنامج إلى نهاية الملف.

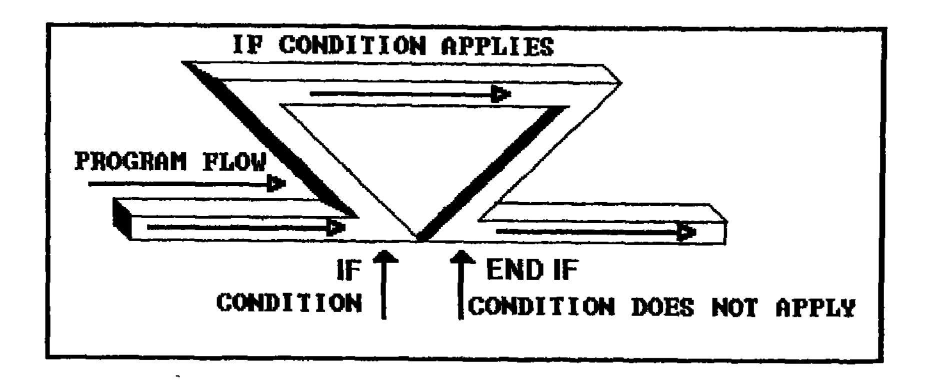
ويلاحظ فى هذه الحلقة إدخال أوامر الحلقة قليلا إلى الداخل. ويسمى ذلك (Indentation) ويستخدم عادة فى البرنامج لتوضيح الأوامر الخاصة بالحلقة التكرارية خاصة عندما تتعدد الحلقات التكرارية وتصبح متداخلة (Nested).

٢٤ - ٧ إستخدام الأمر (IF-ENDIF)

يستخدم هذا الأمر عندما يراد تنفيذ أمر أو مجموعة من الأوامر مرة واحدة عند تحقق شرط معين. حيث يتم اختبار الشرط التالى للأمر (IF) فإذا تحقق يتم تنفيذ الأوامر التالية له أما إذا لم يتحقق فيتم الإنتقال مرة واحدة إلى الأمر (ENDIF) وتنفيذ الأوامر التالية له أى أنه فى هذه الحالة يتخطى الأوامر المحصورة بين الأمرين (IF,ENDIF). أنظر الشكل (٢٤ - ٢).

فمثلا يمكن أن يحتوى برنامج على الأوامر التالية :

```
IF MARK < 50
    ? "Failed"   & & Display the message
ENDIF
```



شكل (٢٤ - ٢)

وفى هذا المثال يتم اختبار الشرط الموجود بعد الأمر (IF) فإذا كانت الدرجة (MARK) أصغر من ٥٠ درجة تظهر رسالة (Failed) وإذا كانت الدرجة أكبر من أو تساوى ٥٠ درجة لاتظهر هذه الرسالة. والحرف (?) هو أمر من أوامر (+ DBase III) يستخدم فى عرض رسائل على الشاشة.

ويمكن زيادة إمكانيات هذا الأمر عن طريق استخدام (ELSE). فيمكن مثلا كتابة الأوامر التالية فى برنامج :

```
IF MARK < 50
   ? "Failed"
ELSE
   ? "Passed"
ENDIF
```

ويؤدى ذلك إلى ظهور رسالة (Failed) فى حالة تحقق الشرط وظهور رسالة (Passed) فى حالة عدم تحقق الشرط. كما يمكن استخدام مجموعة متداخلة (Nested) من أوامر (IF) كما سيتم الإيضاح فيما بعد فى الجزء الخاص بالتداخل (Nesting).

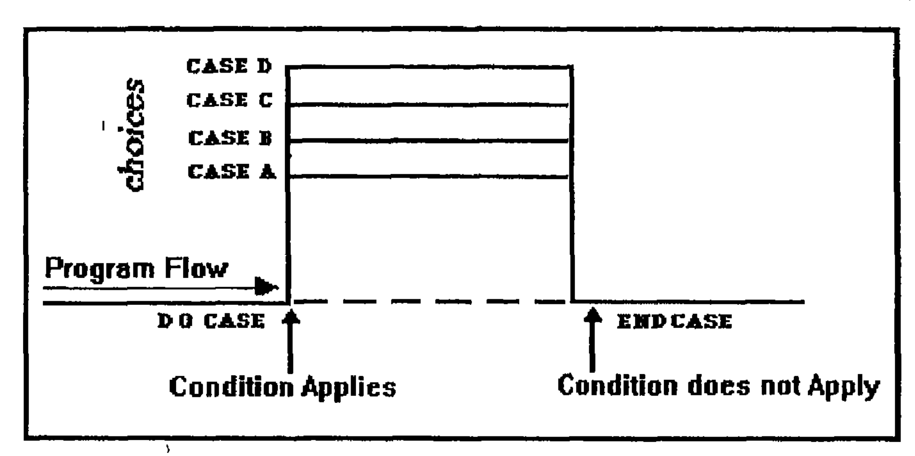
ويمكن استخدام المعاملات المنطقية فى الجمع بين عدة شروط كما يلاحظ من مجموعة الأوامر التالية :

```
IF Age > 40 .AND. .NOT. Married
    DO Payment
ENDIF
```

حيث يتم تنفيذ البرنامج (Payment) فقط للموظفين الذين يزيد عمرهم عن ٤٠ سنة وغير متزوجين. ويجب ملاحظة أن العامل المنطقى يكون محصورا بين نقطتين. كما يجب فصل كل معامل عن المعامل الآخر بمسافة خالية (Space) على الأقل.

٢٤ - ٨ استخدام الأمر (DO CASE - ENDCASE)

عندما يراد التفرع إلى برامج مختلفة بناء على شروط معينة يستخدم الأمر (DO CASE). فعندما يقابل البرنامج الأمر (DO CASE) فإنه يذهب إلى كل سطر يبدا بكلمة (CASE) ويختبر الشرط الموجود معها فإذا تحقق الشرط ينفذ البرنامج أو الأوامر التالية لهذا الشرط وإذا لم يتحقق الشرط فإنه يذهب إلى باقى السطور التى تبدأ بكلمة (CASE) ويختبر الشروط الخاصة بها. ويستخدم هذا الأمر بصفة خاصة فى القوائم حيث يتم التفرع من البرنامج الرئيسى إلى برامج فرعية بناء على اختيار المستخدم. انظر الشكل (٢٤ - ٣)



شكل (٢٤ - ٣)

وتنتهى هذه المجموعة بالأمر (ENDCASE) الذى يؤدى إلى الإنتقال إلى الأوامر التى تلى هذا الأمر.

كما يمكن استخدام الأمر (OTHERWISE) مع هذا الأمر لتنفيذ أمر معين فى حالة عدم تحقق أى شرط من الشروط التى تلى كل أمر (CASE).

ولتوضح ذلك يمكن ملاحظة الأوامر التالية :

```
DO CASE
    CASE Choice = "A"
        DO MEdits
    CASE Choice = "B"
        DO MDel
    CASE Choice = "C"
        DO MRep
    OTHERWISE
        DO Leave
ENDCASE
```

وفى هذا المثال يتم تنفيذ برنامج (MEdit) عند اختيار المستخدم للحرف (A) ويتم تنفيذ برنامج (MDel) عند اختيار المستخدم للحرف(B) ويتم تنفيذ برنامج (MRep) عند اختيار المستخدم للحرف (C) ويتم تنفيذ برنامج (Leave) عند اختيار المستخدم لأى حرف آخر غير الحروف (A,B,C).

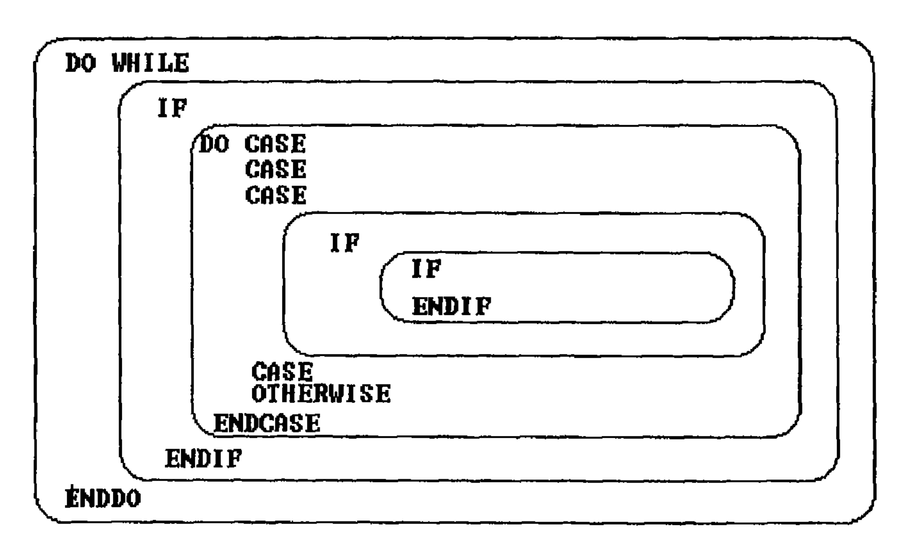
٢٤ - ٩ التداخل (Nesting)

يمكن استخدام أوامر التحكم السابق شرحها بالتداخل فيما بينها وذلك كالآتى مثلا :

```
IF MARRIED
    DO CASE
        CASE Age > 20 .AND. Age < 30
        ------
        ------      Commands
        ------
        CASE Age > 30 .AND. Age < 40
```

```
            ------
            ------      Commands
            ------
      ENDCASE
   ENDIF
```

ويلاحظ فى هذا المثال أن الحلقة الخاصة بالأمر (DO CASE) داخله بالكامل فى الحلقة الخاصة بالأمر (IF). ويلاحظ أيضا فى هذه الحالة أن الأمر (ENDCASE) يجب أن يسبق الأمر(ENDIF) وإلا يصبح التداخل غير صحيح. كما يفيد إدخال الكتابة إلى الداخل (Indentation) فى توضيح حدود كل حلقة. أنظر الشكل (٢٤ - ٤).



شكل (٢٤ - ٤)

ملاحظة

يجب دائما التأكد أن كل أمر من الأوامر (DO WHILE, IF, DO CASE) له أمر إنهاء خاص به (ENDDO, ENDIF, ENDCASE) على الترتيب. كما يجب التأكد من عدم إختلاف ترتيب أوامر الإنهاء عن ترتيب أوامر البداية. ويجب ملاحظة أن أوامر الإنهاء تكتب كلمة واحدة وليست كلمتين أى (ENDDO, ENDIF, ENDCASE) بدون مسافات بينها.

ويمكن استخدام الملاحظات فى تحديد أى أمر نهاية يتبع أى أمر بداية. ويمكن توضيح ذلك من البرنامج التالى :

```
DO WHILE .NOT. EOF()
   LIST Name , Age
   SKIP
ENDDO  && while not EOF()
```

هذه الملحوظة التى تلى الحرفين (&&) تؤدى إلى توضيح أن أمر الإنهاء (ENDDO) يختص بأمر البداية المحدد فى الملحوظة.

٢٤ - ١٠   إستخدام الأمر (LOOP)

يستخدم الأمر (LOOP) للرجوع من أى مكان داخل الحلقة التكرارية إلى أول الحلقة دون إستكمال أوامر الحلقة. ويمكن توضيح ذلك من خلال الأوامر التالية :

```
DO WHILE .NOT. EOF()
   IF Mark < 50
      ? "FAILED"
      SKIP
      LOOP
   ENDIF &&  Mark < 50
      ----------
      ----------
      ----------       Commands
      ----------
      ----------
ENDDO && WHILE .NOT. EOF()
```

وفى هذا المثال يتم اختبار درجة الطالب فإذا كانت أقل من ٥٠ درجة تظهر الرسالة (FAILED) ويتم الإنتقال إلى السجل التالى. ويؤدى الأمر (LOOP) إلى الرجوع إلى الحلقة لاختبار درجة الطالب الجديد. ويستمر ذلك حتى يتم الوصول إلى أى طالب درجته

أكبر من ٥٠ درجة حيث يتخطى البرنامج مجموعة الأوامر المحصورة بين (IF, ENDIF) وينفذ مجموعة الأوامر التالية. وعندما يصل البرنامج إلى الأمر (ENDDO) يتم الإنتقال إلى أول الحلقة التكرارية مرة أخرى وتتكرر هذه العملية.

٢٤ - ١١ الخروج من الحلقة التكرارية

يستخدم الأمر (EXIT) فى الخروج من الحلقة التكرارية والإنتقال إلى الأوامر التى تلى الأمر (ENDDO). وهو يختلف عن الأمر (LOOP) حيث أن الأمر (LOOP) يؤدى إلى الرجوع إلى أول الحلقة التكرارية والبدء فى تنفيذها من جديد. أما الأمر (EXIT) فإنه لايؤدى إلى الرجوع إلى أول الحلقة التكرارية ولكنه يؤدى إلى الخروج منها وتنفيذ الأوامر التى تلى الأمر (ENDDO).

ملاحظة

ما سبق ذكره فى هذا الفصل ينطبق أيضا على كل برامج عائلة (DBase) مثل (DBase IV) ، (FoxBase +) ، (FoxPro).



الفصل الخامس والعشرون

متغيرات الذاكرة

(Memory Variables)



متغيرات الذاكرة هى متغيرات من نوع خاص يتم فيها تخزين البيانات تخزينا مؤقتا خارج ملف قاعدة البيانات. وعن طريق هذه المخازن المؤقتة يمكن إجراء العمليات الحسابية أو أى عمليات على بيانات الملف دون تغيير بيانات الملف مباشرة.

ويجب ألايزيد طول إسم المتغير عن عشرة حروف ويمكن أن تحتوى على حروف أو أعداد أو شرطة سفلية (Underscore) ولكن يجب أن تبدأ بحرف ولاتحتوى على مسافات. وهناك أربعة أنواع من متغيرات الذاكرة وهى متغيرات حرفية (Character) ومتغيرات تاريخية (Date) ومتغيرات عددية (Numeric) ومتغيرات منطقية (Logical) ولكن ليست هناك متغيرات لتخزين الملاحظات (Memo).

ويمكن أن يصل عدد المتغيرات المستخدمة فى البرنامج إلى ٢٥٦ متغيرا بحيث لايزيد حجمها الكلى عن ٦٠٠٠ حرفا. ويمكن زيادة حجمها بكتابة الأمر (MVARSIZ) فى ملف المواصفات (Config.sys) يليه عدد الحروف المراد استخدامها وذلك حسب الذاكرة المؤقتة المتاحة فى الجهاز.

ملاحظة

عندما تكون هناك حقول فى ملف قاعدة البيانات لها نفس الإسم مثل متغيرات الذاكرة فإن الأسبقية تكون لحقول الملف عندما يشار إلى هذا الإسم. فمثلا عندما يكون هناك حقل مثل (Name) فى ملف قاعدة البيانات وتم إنشاء متغير ذاكرة بنفس الإسم فإن أى بيانات يتم إدخالها إلى المتغير (Name) تذهب إلى الحقل وليس إلى متغير الذاكرة. وللتغلب على ذلك يتم إضافة الحرف (M) قبل إسم المتغير حتى يصبح مختلفا عن إسم الحقل وفى نفس الوقت يعطى معنى محتويات الحقل أى يصبح (MName).

ويراعى أيضا فى اختيار أسماء متغيرات الذاكرة أن تكون مختلفة عن الأسماء المحجوزة لبرنامج (+ DBase III) مثل (Continue) مثلا. إرجع إلى الجزء الرابع من الكتاب للتعرف على جميع الأوامر والدوال المستخدمة فى برنامج (+DBase III).

٢٥ - ١ أنواع متغيرات الذاكرة (Memory Variables)

كما سبق الإيضاح هناك أربعة أنواع من متغيرات الذاكرة ويتم شرحها فى الأجزاء التالية.

٢٥ - ١ - ١ المتغيرات الحرفية (Character)

وتستخدم لتخزين المدخلات الحرفية ويمكن إدخال بيانات حرفية تحتوى على كود الآسكى الخاص بالحروف باستخدام الدالة (CHR) كما سيتم الإيضاح فيما بعد.

والمتغيرات الحرفية يمكن أن تحتوى على عدد حروف بحد أقصى ٢٥٤ حرفا وتحتل فى الذاكرة مساحة تخزينية تساوى عدد الحروف المخزنة مضافا إليه حرفين.

٢٥ - ١ - ٢ المتغيرات التاريخية (Date)

وهى تستخدم لتخزن التاريخ وحجم المتغير التاريخى ٨ حروف دائما. ويتم إدخال التاريخ بالطريقة الأمريكية (mm/dd/yy) أى الشهر ثم اليوم ثم السنة كما يمكن استخدام صورة أخرى باستخدام الأمر (SET DATE) كما .سيتم الإيضاح فيما بعد.

ويمكن إجراء عمليات حسابية على التاريخ مثل طرح تاريخ من تاريخ للحصول على عدد الأيام المحصورة بينهما. كما يمكن طرح عدد الأيام من تاريخ معين للحصول على تاريخ آخر.

٢٥ - ١ - ٣ المتغيرات العددية (Numeric)

وتستخدم لتخزين الأعداد التى يمكن إجراء عمليات حسابية عليها ويمكن أن تحتوى على ١٥ رقما بما فيها الأرقام العشرية (Decimal) التى يجب ألا يزيد عددها عن ٩ أرقام عشرية.

٢٥ - ١ - ٤ المتغيرات المنطقية (Logical)

وهى متغيرات يتم فيها تخزين حرف واحد فقط يمثل حالة البيان إذا كان صحيحا (True) أو غير صحيح (False). هذا الحرف يكون أحد الحروف التالية .(T, F, Y, N)

٢٥ - ٢ إنشاء متغيرات الذاكرة

يتم إنشاء متغيرات الذاكرة بمجرد تخزين بيانات فيها ويستخدم لذلك الأمر(Store). كما أن البرنامج يقوم بتحديد نوع المتغير حسب نوع البيانات التى يتم إدخالها.

ويمكن إنشاء متغيرات الذاكرة بطريقة أخرى وذلك بكتابة إسم المتغير أولا وبعده علامة التساوى ثم البيانات المطلوب تخزينها فيه. وفى الأجزاء التالية يتم عرض عدة أمثلة توضح طريقة إنشاء متغيرات الذاكرة بالطريقتين.

٢٥ - ٢ - ١ إنشاء المتغيرات المنطقية (Logical Variables)

الأمر التالى يؤدى إلى إنشاء متغير يسمى (Married) ويتم تخزين القيمة (.T.) أى (True) فيه :

```
STORE .T. To Married
```

كما يمكن استخدام الأمر التالى ليحقق نفس النتيجة :

```
Married = .T.
```

وفى هذه الحالة تصبح البيانات يمين علامة التساوى هى محتويات المتغير ويمكن تغيير هذه المحتويات بعد ذلك حسب الحاجة.

٢٥ - ٢ - ٢ إنشاء المتغيرات الحرفية (Character Variables)

الأمر التالى يؤدى إلى إنشاء متغير ذاكرة إسمه (mname) ويتم تخزين الإسم (Mohamed) فيه :

```
STORE "Mohamed" TO mname
```

ويلاحظ فى هذه الحالة وضع الحروف (String) بين علامات تنصيص (Quotation) أما إذا كانت الحروف المطلوب إدخالها فى المتغير تحتوى على

علامات تنصيص داخلها يتم استخدام علامات تنصيص مختلفة عن العلامات داخل الحروف (String). فمثلا يمكن استخدام الأمر التالى :

```
STORE "That's incorrect..try again" TO mmessage
```

ويمكن إنشاء متغير ذاكرة لايحتوى على أى بيانات باستخدام الدالة (Space ()) كالآتى :

```
STORE SPACE(20) TO mname
```

وهذا يؤدى إلى حجز متغير إسمه (mname) طوله ٢٠ حرفا وليس فيه أى بيانات. وذلك حتى يمكن استخدامه بعد ذلك فى تخزين بيانات الإسم.

٢٥ - ٢ - ٣ إنشاء المتغيرات التاريخية (Date Variables)

لإنشاء متغير تاريخى يمكن استخدام دالة التحويل من الحروف إلى التاريخ (Character to date conversion function) التى تختصر إلى (CTOD ( )). هذه الدالة تؤدى إلى تحويل الحروف التى يكتبها المستخدم ممثلة تاريخا معينا إلى قيمة مقابلة يخزنها البرنامج. فمثلا لإنشاء متغير تاريخى إسمه (Birthday) يتم كتابة الأمر التالى :

```
birthday = CTOD ('20/1/49')
```

ويمكن بعد ذلك تغيير محتويات هذا المتغير بأى تاريخ آخر. كما يمكن إنشاء متغير تاريخى ليس به أى تاريخ بكتابة الأمر التالى :

```
birthday = CTOD('/ /')
```

كما يمكن استخدام الأمر (STORE) فى إنشاء المتغير التاريخى كالآتى :

```
STORE CTOD('/ /') To birthday
```

كما يمكن استخدام دالة التاريخ (DATE ()) فى إدخال تاريخ اليوم الحالى كالآتى :

```
STORE DATE () TO today
```

٢٥ - ٢ - ٤ إنشاء المتغيرات العددية (Numeric Variables)

لإنشاء متغيرات عددية يكفى إدخال صفر فى المتغير كالآتى مثلا :

```
STORE 0 TO number
```

مع ملاحظة أن هذا المتغير يقبل رقما صحيحا فقط. أما إذا أريد إنشاء متغير يقبل كسرا عشريا فيتم وضع نقطة الكسر العشرى (Decimal Point) ، وذلك كالآتى :

```
STORE 0.00 TO number
```

فى هذه الحالة يتم إنشاء متغير عددى يقبل رقمين عشريين. ويمكن عن طريق زيادة عدد الأصفار بعد العلامة العشرية زيادة عدد الأرقام العشرية التى يقبلها المتغير العددى.

كما يمكن إنشاء عدة متغيرات عددية من نفس النوع باستخدام أمر واحد. وذلك كالآتى مثلا :

```
STORE 0.00 TO num1 , num2 , num3
```

ملاحظة

عند اختيار إسم متغير الذاكرة يراعى أن يكون معبرا عن محتويات هذا المتغير. كما يراعى أيضا عندما يكون إسم المتغير مطابقا لإسم حقل فى ملف قاعدة البيانات أن يضاف الحرف (M) قبل إسم المتغير حتى يتم تمييزه عن إسم الحقل. فمثلا إذا كان هناك حقل إسمه (Name) فى قاعدة البيانات يمكن إنشاء متغير ذاكرة إسمه (MName) أو (M_Name).

٢٥ - ٣ أهمية متغيرات الذاكرة

تستخدم متغيرات الذاكرة كما سبق الإيضاح كمخزن مؤقت للبيانات التى يتم إدخالها إلى ملف قاعدة البيانات أو إخراجها إلى أجهزة الإخراج المختلفة. وهذه العملية فى منتهى الأهمية فى كتابة البرامج وذلك لأنها تعطى الفرصة للمستخدم لاختبار البيانات التى يقوم بإدخالها قبل تخزينها فى قاعدة البيانات. فمثلا عندما يراد إدخال بيانات فى حقل الإسم (Name) فى قاعدة بيانات معينة يتم اتباع الخطوات التالية :

١ - يتم إنشاء متغير ذاكرة حرفى نسميه مثلا (MName) بحيث يكون بنفس طول حقل الإسم (Name) الخاص بملف قاعدة البيانات.
٢ - يتم استخدام هذا المتغير فى تخزين الإسم الذى يقوم المستخدم بإدخاله تخزينا مؤقتا.
٣ - يتم عرض الإسم الذى أدخله المستخدم حتى يتأكد أنه الإسم المطلوب ويتم سؤال المستخدم إذا كان الإسم صحيحا أم لا.
٤ - إذا كان رد المستخدم (No) يتم إعطاؤه الفرصة لتصحيح الإسم.
٥ - إذا كان رد المستخدم (Yes) يتم استخدام الأمر (REPLACE) لاستبدال محتويات الحقل بالإسم الذى تم تخزينه فى متغير الذاكرة.
٦ - يتم تكرار هذه العملية مع الأسماء الأخرى التى يتم إدخالها. وفى كل مرة تستبدل محتويات متغير الذاكرة بالإسم الجديد الذى يتم إدخاله. وهكذا يلاحظ أن متغير الذاكرة (Memory Variable) يعمل كحلقة إتصال بين المستخدم وملف قاعدة البيانات.

ويمكن استخدام متغير الذاكرة أيضا مع المخرجات (Output) التى يتم توجيهها إلى الشاشة أو الطابعة بنفس الطريقة التى سبق شرحها.

٢٥ - ٤ المتغيرات العامة والمتغيرات الخاصة

يتم تصنيف متغيرات الذاكرة إلى متغيرات عامة (Public Variables) ومتغيرات خاصة (Private Variables). والمتغير العام (Public) هو المتغير الذى يتم إنشاؤه فى أى برنامج فرعى أو رئيسى ويكون مؤثرا فى جميع البرامج الأخرى. أما المتغير الخاص (Private) فهو المتغير الذى يتم إنشاؤه فى أى برنامج فرعى ولايؤثر إلا فى هذا البرنامج أو البرامج الفرعية المتفرعة منه. وفى الأجزاء التالية يتم توضيح خصائص كل من النوعين بشىء من التفصيل.

٢٥ - ٤ - ١ المتغيرات العامة (Public Variables)

يتم إنشاء المتغير العام فى خطوتين الأولى إعلان هذا المتغير كمتغير عام (Public) والثانية إنشاء هذا المتغير. وذلك كالآتى مثلا :

```
PUBLIC Average
Average = 0.00
```

حيث يتم إعلان (Declare) هذا المتغير كمتغير عام (Public) كما يتم إنشاؤه كمتغير عددى (Numeric) يسمح بإدخال رقمين عشريين. وفى هذه الحالة يصبح المتغير (Average) متغيرا عاما أى أنه يستخدم فى جميع البرامج المستخدمة فى هذا الوقت. سواء كانت برامج رئيسية أو فرعية.

٢٥ - ٤ - ٢ المتغيرات الخاصة (Private Variables)

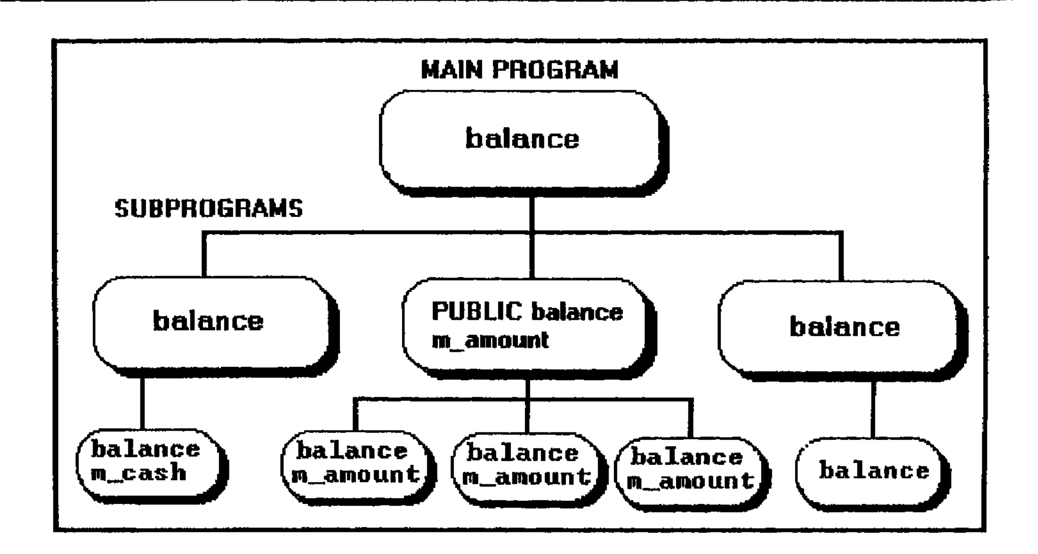
متغير الذاكرة بصفة عامة يصبح خاصا إذا لم يتم إعلانه كمتغير عام (Public). أى أنه يكفى إنشاء المتغير فقط لكى يصبح متغيرا خاصا. فمثلا الأمر التالى يؤدى إلى إنشاء متغير خاص فى برنامج فرعى معين :

```
Mcost = 0.00
```

فى هذه الحالة يصبح متغير الذاكرة (Mcost) متغيرا خاصا (Private) فى هذا البرنامج الفرعى والبرامج الفرعية المتفرعة منه.

والشكل (٢٥ - ١) يوضح العلاقة بين المتغيرات العامة(Public Variables) والمتغيرات الخاصة (Private Variables).

ويلاحظ فى هذا الشكل أن المتغير العام (balance) يؤثر فى جميع البرامج رغم إنشائه داخل برنامج فرعى (Subprogram) أما المتغير (m_amount) فإنه يؤثر فقط فى البرامج الفرعية المتفرعة من البرنامج الذى تم إنشاؤه خلاله. بينما يعتبر المتغير (m_cash) متغيرا خاصا ببرنامج فرعى واحد ولايؤثر فى باقى البرامج.



شكل (٢٥ - ١)

ملاحظة

أى متغيرات ذاكرة خاصة يتم إنشاؤها فى البرنامج الرئيسى (Main Program) تؤثر فى جميع البرامج الفرعية وبالتالى تصبح مثل المتغيرات العامة تماما. ولذلك يفضل دائما إنشاء المتغيرات التى يراد استخدامها فى جميع البرامج الفرعية مرة واحدة فى البرنامج الرئيسى بدلا من الحاجة إلى تكرار إنشاء متغيرات الذاكرة الخاصة فى كل برنامج فرعى.

٢٥ - ٥ التخلص من متغيرات الذاكرة

من المهم جدا التخلص من متغيرات الذاكرة بعد إنتهاء البرنامج حتى لاتؤثر فى أى برامج أخرى يراد تشغيلها. كما يراد أحيانا التخلص من بعض هذه المتغيرات أثناء تنفيذ البرنامج. والمتغيرات الخاصة (Private) تختفى بمجرد انتهاء البرنامج أو انتهاء البرنامج الفرعى الذى تم إنشاؤها خلاله. وهناك طريقة أخرى للتخلص من متغير ذاكرة خاص أثناء تنفيذ البرنامج وذلك باستخدام الأمر (RELEASE) وذلك كالآتى مثلا :

```
RELEASE maverage
```

كما يمكن التخلص من مجموعة من المتغيرات الخاصة بأمر واحد كالآتى :

RELEASE ALL LIKE m*

ويلاحظ هنا إستخدام الحرف الشامل (*) وذلك للتخلص من جميع المتغيرات الخاصة التى تبدأ بالحرف (m).

كما يمكن التخلص من كل المتغيرات الخاصة ماعدا بعض هذه المتغيرات وذلك كالآتى مثلا :

RELEASE ALL EXCEPT m*

فى هذه الحالة يتم التخلص من جميع المتغيرات الخاصة ماعدا المتغيرات التى تبدأ بالحرف (m).

اما المتغيرات العامة (PUBLIC) فهى لاتختفى باختفاء البرنامج بل تظل موجودة فى الذاكرة. لذلك يلزم التخلص منها فى نهاية البرنامج باستخدام الأمر (RELEASE) أو (CLEAR MEMORY) أو (CLEAR ALL).

والأمر (RELEASE) يتم بواسطته التخلص من متغير عام محدد ولاتستخدم الحروف الشاملة (Global Characters) فى هذه الحالة. ولكن يمكن التخلص من عدة متغيرات بكتابة أسمائها وبينها فاصلة (Comma) وذلك كالآتى مثلا :

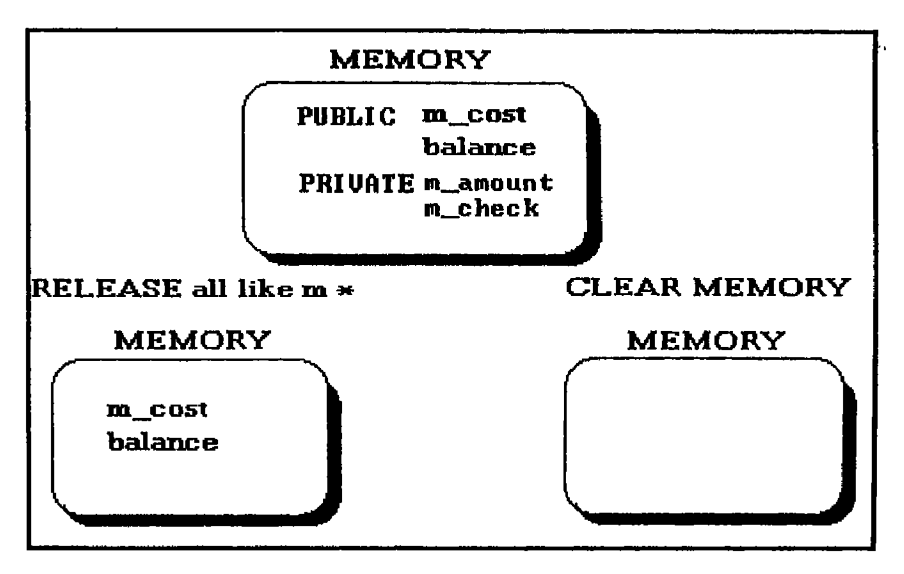
RELEASE mcost, mamount

أما الأمر (CLEAR ALL) والأمر (CLEAR MEMORY) فيستخدمان للتخلص من جميع المتغيرات العامة التى تم إنشاؤها خلال البرنامج.

ملاحظة

المتغيرات العامة بصفة خاصة يلزم التخلص منها خلال البرنامج لأنها تظل موجودة فى الذاكرة بعد انتهاء تشغيل البرنامج وقد تؤثر على تشغيل أى برنامج آخر.

والشكل (٢٥ - ٢) يوضح كيف يتم التخلص من متغيرات الذاكرة العامة والخاصة.



شكل (٢٥ - ٢)

٢٥ - ٦ ملفات الذاكرة (Memory Files)

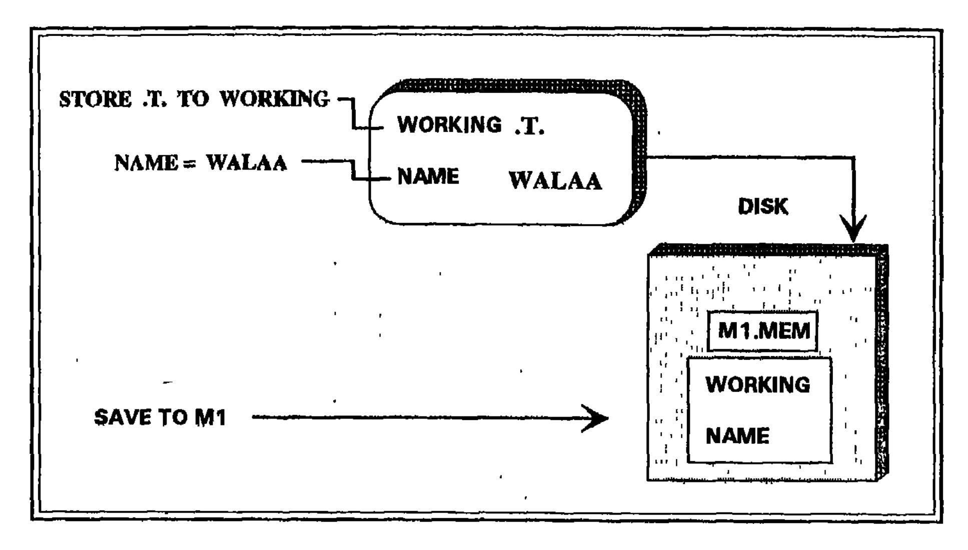
يمكن تخزين متغيرات الذاكرة فى ملف ذاكرة (Memory File) وذلك حتى يمكن استخدام هذه المتغيرات فى أى برنامج دون الحاجة إلى إنشاء المتغيرات مرة ثانية. ويستخدم الأمر (SAVE) لتخزين محتويات هذه المتغيرات فى ملف الذاكرة. كما يستخدم الأمر (RESTORE) لاسترجاع هذه المتغيرات فى الذاكرة مرة ثانية.

ولإنشاء ملف الذاكرة يتم أولا إنشاء متغيرات الذاكرة من مشيرة النقطة (Dot Prompt) ثم يتم استخدام الأمر (SAVE) مع إسم الملف المطلوب إنشاؤه ويقوم البرنامج بإضافة الإمتداد (mem.) إلى إسم الملف.

فمثلا عند إنشاء عدة متغيرات ذاكرة ويراد تخزينها فى ملف ذاكرة إسمه (M1) يستخدم الأمر التالى :

```
SAVE TO M1
```

والشكل (٢٥ - ٣) يوضح عملية إنشاء ملف ذاكرة



شكل (٢٥ - ٣)

٢٥ - ٧ إسترجاع ملفات الذاكرة

يستخدم الأمر (RESTORE) لاسترجاع ملفات الذاكرة (Memory Files) فى ذاكرة الحاسب المؤقتة (RAM) حتى يمكن استخدام المتغيرات المخزنة فيها وذلك كما يلى :

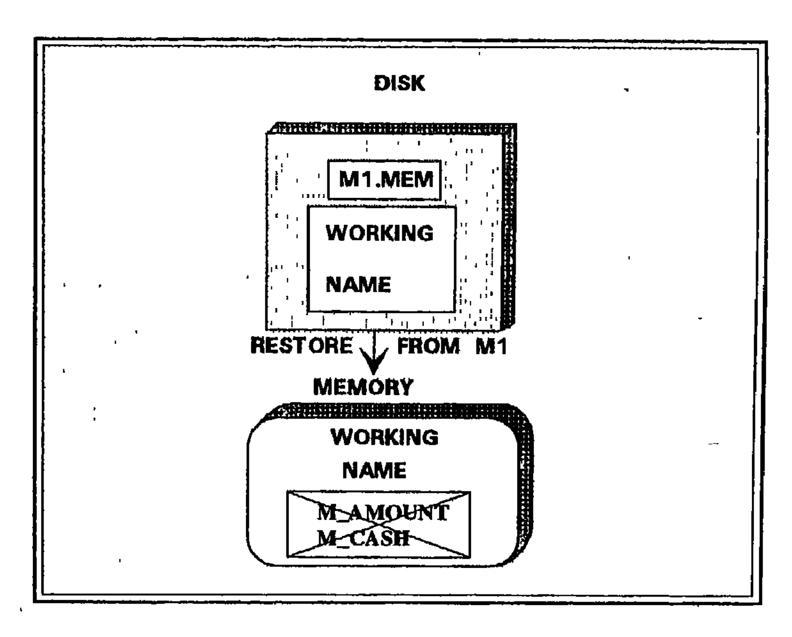
```
RESTORE FROM M1
```

ويجب ملاحظة أن استرجاع أى ملف ذاكرة يؤدى إلى مسح كل متغيرات الذاكرة الموجودة فى الذاكرة المؤقتة للحاسب وذلك إذا لم يتم تنبيه البرنامج إلى المحافظة على المتغيرات الموجودة فى الذاكرة ويستخدم الأمر (ADDITIVE) لعمل ذلك. حيث أنه يؤدى إلى إضافة المتغيرات الموجودة فى ملف الذاكرة إلى المتغيرات الموجودة فى الذاكرة المؤقتة.

فمثلا لكى يتم استرجاع الملف (M1.mem) مع الإحتفاظ بالمتغيرات الموجودة فى الذاكرة المؤقتة يستخدم الأمر التالى :

RESTORE FROM M1 ADDITIVE

والشكل (٢٥ - ٤) يوضح عملية استرجاع ملف الذاكرة باستخدام الأمر (RESTORE) مع فرض وجود متغيرات فى الذاكرة المؤقتة.



شكل (٢٥ - ٤)

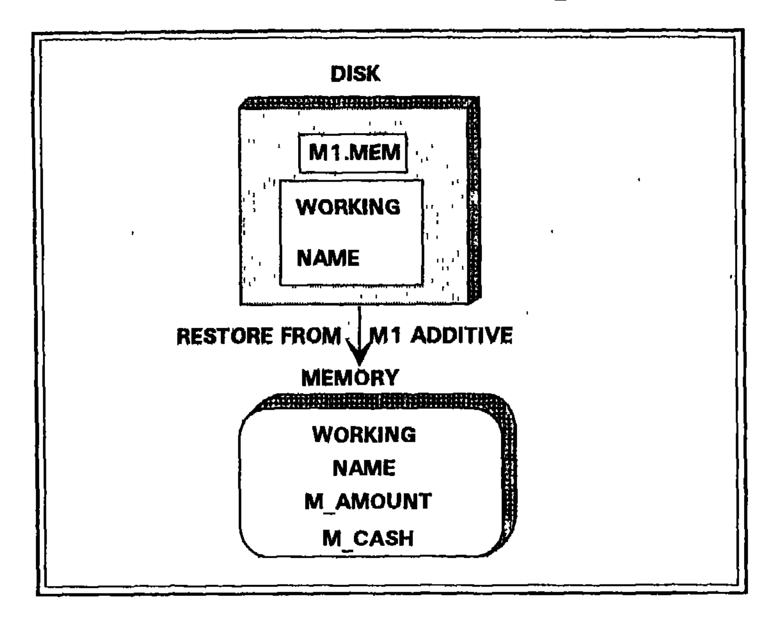
ويلاحظ اختفاء المتغيرات الموجودة أصلا فى الذاكرة لتحل محلها متغيرات ملف الذاكرة (M1.mem).

أما الشكل (٢٥ - ٥) فيوضح استخدام الأمر (ADDITIVE) لاستعادة نفس الملف. ويلاحظ فى هذه الحالة إضافة متغيرات ملف الذاكرة إلى المتغيرات الموجودة فى الذاكرة المؤقتة.

ملاحظة

كل متغيرات الذاكرة التى يتم إنشاؤها من مشيرة النقطة (Dot Prompt) تكون عامة (Public) ولكن عند استرجاعها بواسطة الأمر (RESTORE) فإن حالتها - أى اذا كانت عامة أو خاصة - تتوقف على المكان الذى يتم استرجاعها فيه. فإذا استرجعت

عند مشيرة النقطة (Dot Prompt) فإنها تصبح عامة (Public) وإذا استرجعت فى البرنامج الرئيسى فإنها تصبح عامة أيضا.



الشكل (٢٥ - ٥)

أما إذا استرجعت فى برنامج فرعى فإنها تصبح خاصة (Private) لهذا البرنامج الفرعى والبرامج الفرعية المتفرعة منه فقط. أما إذا أريد استرجاع بعض المتغيرات فى برنامج فرعى بحيث تكون متغيرات عامة (Public) فيجب إعلانها أولا عامة قبل استرجاع ملف الذاكرة.

فمثلا ملف الذاكرة (Accounts.mem) يحتوى على ثلاثة متغيرات (mcost, mname, mamount) ويراد استرجاع هذا الملف داخل برنامج فرعى بحيث تكون هذا المتغيرات عامة (Public). فلتنفيذ ذلك يتم كتابة السطرين التاليين :

```
PUBLIC mcost, mname, mamount
RESTORE FROM Accounts
```

فى هذه الحالة تصبح هذه المتغيرات عامة (Public).

٢٥ - ٨ أهمية استخدام ملفات الذاكرة

توفر ملفات الذاكرة (Memory Files) لمخطط البرامج المحترف مرونة عالية فى التحكم فى البرنامج حيث يمكنه تخزين عدة ملفات ذاكرة كل منها يختص بوظيفة معينة فى البرنامج ويمكنه استرجاع الملف المطلوب لأداء هذه الوظيفة فى البرنامج. كما يمكنه استرجاع ملف آخر خاص بوظيفة أخرى مع ملاحظة أنه يمكنه التخلص من الملف السابق بمجرد استرجاع الملف الجديد باستخدام الأمر (RESTORE) كما يمكنه الإحتفاظ بالملف السابق فى الذاكرة باستخدام الأمر (ADDITIVE) حسب الحاجة. كما يمكنه أيضا تعديل محتويات ملف الذاكرة أثناء كتابة البرنامج وذلك بإضافة متغيرات جديدة أو حذف متغيرات مخزنة به.

فمثلا عندما يحتاج مخطط البرامج إلى إضافة متغير ذاكرة جديد إلى ملف الذاكرة فإنه يقوم باسترجاع ملف الذاكرة من مشيرة النقطة (Dot Prompt) ثم يتم تخزين المتغير الجديد فى الذاكرة باستخدام الامر (SAVE). فى هذه الحالة يتم تخزين متغيرات الذاكرة الموجودة فى الذاكرة المؤقتة متضمنة المتغير الذى تمت إضافته.

ملاحظة

ماسبق ذكره فى هذا الفصل ينطبق أيضا على كل برامج عائلة (DBase) مثل (DBase IV) ، (FoxBase) ، (FoxBase +) ، (FoxPro).

الفصل السادس والعشرون

أوامر التجهيز فى البرنامج الرئيسى


Converted by Tiff Combine - (no stamps are applied by registered version)

البرنامج الرئيسى (Main Program) هو البرنامج الذى يستدعى ويشغل البرامج الفرعية كما يجهز محيط التشغيل (Working Environment) للبرنامج كله والذى يشمل تحديد وحدة الأقراص المستخدمة أو الفهرس الفرعى المستخدم وتحديد حالة الشاشة بالإضافة إلى وظائف أخرى متعددة سيتم شرحها فيما بعد. كما يقوم البرنامج الرئيسى عادة بعرض القائمة الرئيسية التى يقوم المستخدم بالإختيار منها. كما يقوم البرنامج الرئيسى بفتح ملفات قاعدة البيانات والملفات المصاحبة لها فى بداية البرنامج كما يقوم عادة بإغلاق هذه الملفات فى نهاية البرنامج.

٢٦ - ١ تركيب البرنامج الرئيسى

كما سبق الإيضاح فإن البرنامج الرئيسى يمكن تقسيمه إلى أربعة أقسام وهى المقدمة وأوامر التجهيز (Set up) وأوامر البرنامج وأوامر الخروج وقد تم فيما سبق توضيح محتويات كل قسم. وفى الأجزاء التالية يتم إلقاء مزيد من الضوء على أوامر التجهيز (Set up) التى تكتب عادة فى البرنامج الرئيسى لتكون مؤثرة على جميع البرامج المتفرعة منه.

٢٦ - ٢ أوامر التجهيز (Set Up)

تبدأ أوامر التجهيز بإغلاق جميع ملفات قواعد البيانات والتخلص من جميع متغيرات الذاكرة وذلك لأن هناك احتمالا أن يكون أحد ملفات قواعد البيانات مفتوحا أو تكون هناك متغيرات ذاكرة موجودة فى الذاكرة المؤقتة (RAM). ويتم ذلك باستخدام الأمر (CLEAR ALL).

ويتم بعد ذلك تحديد بيانات محيط التشغيل (Working Environment) والتى يتم عن طريقها تحديد حالة الشاشة وعمود الحالة (Status Bar) ووحدة الأقراص المستخدمة ... الخ.

٢٦ - ٢ - ١ تحديد بيانات محيط التشغيل

عند بداية تشغيل البرنامج يكون هناك وضع مبدئى (Default) لمحيط التشغيل. هذا الوضع المبدئى يشمل مثلا ظهور عمود الحالة (Status bar) أسفل الشاشة بالإضافة إلى باقى الرسائل التى تظهر على الشاشة وأشياء أخرى متعددة سيتم

دراستها فى هذا الجزء.

وبالنسبة لمخطط البرامج فإنه يكون مخيرا بين الإحتفاظ بهذا الوضع المبدئى أو تغييره أو تغيير جزء منه فقط. ويستخدم الأمر (SET) لتنفيذ ذلك مع إضافة المعامل المناسب كما سيتم الإيضاح.

٢٦ - ٢ - ٢ إستخدام الأمر (SET TALK)

الوضع المبدئى لهذا الأمر هو (ON) وهو يعنى ظهور خطوات تشغيل البرنامج على الشاشة أثناء تنفيذه. واذا أراد مخطط البرامج تغيير هذا الوضع فإنه يكتب الأمر (SET TALK OFF) فى بداية البرنامج. فمثلا عند كتابة الأمر التالى من مشيرة النقطة (Dot Prompt) يلاحظ ظهور السطر الذى يليه.

```
.STORE "Enter cadet name" TO mname
Enter cadet name
```

ويحدث نفس الشئ بالنسبة لباقى أوامر البرنامج مما يؤدى إلى ظهور رسائل على الشاشة ليس مطلوبا ظهورها أثناء تنفيذ البرنامج. ولذلك يتم كتابة الأمر (SET TALK OFF) عادة فى بداية البرنامج الرئيسى. وفى هذه الحالة يصبح هذا الأمر مؤثرا على جميع البرامج الفرعية بالإضافة إلى البرنامج الرئيسى.

٢٦ - ٢ - ٣ إستخدام الأمر (SET ESCAPE)

عند الضغط على مفتاح الهروب (Esc) أثناء تشغيل البرنامج فإن البرنامج يتوقف. فإذا أراد مخطط البرامج أن يمنع حدوث ذلك فإنه يكتب الأمر (SET ESCAPE OFF). وهذا الأمر يؤدى إلى إيقاف وظيفة مفتاح الهروب (Esc) أثناء تشغيل البرنامج.

واستخدام هذا الأمر يساعد مخطط البرامج على التحكم فى طريقة ووقت الخروج من البرنامج. حيث أن الخروج الفجائى فى أى وقت قد يسبب متاعب كثيرة نتيجة عدم التأكد من إغلاق جميع الملفات قبل الخروج.

٢٦ - ٢ - ٤ إستخدام الجـرس (Bell)

الوضع المبدئى للبرنامج هو تشغيل الجرس (Bell) عندما يمتلىء الحقل بالبيانات أو عندما يدخل المستخدم مدخلات خطأ. وقد يكون مطلوبا التحكم فى هذا الجرس أثناء تشغيل البرنامج وذلك بجعله قاصرا على أخطاء معينة للمستخدم. ويتم إلغاء الجرس باستخدام الأمر (SET BELL OFF) كما يمكن إعادته باستخدام الأمر (SET BELL ON).

٢٦ - ٢ - ٥ إستخدام الألوان (Colors)

يمكن التحكم فى ألوان الشاشة سواء كانت ألوان الأرضية (Background) أو الأعمدة الضوئية (Highlights) المثلة للحقول أو الكتابة داخل هذه الأعمدة الضوئية. ويتم ذلك باستخدام الأمر (SET COLOR TO) ثم تحديد الألوان المطلوبة لكل منطقة من المناطق التى سبق ذكرها.

كما يمكن تغيير حالة الشاشة من ألوان إلى أبيض وأسود والعكس ويتم ذلك باستخدام الأمر (SET COLOR ON/OFF) حيث يتم التغيير بين (ON) و (OFF). كما يمكن استخدام الدالة (ISCOLOR()) لاختبار الشاشة إذا كانت ملونة أو غير ملونة. وبناء على ذلك يتم التغيير بين الألوان والأبيض والأسود حسب الحاجة.

فمثلا يمكن أن يتضمن البرنامج السطور التالية :

```
IF ISCOLOR ()
    SET COLOR ON
ENDIF
```

فإذا كانت قيمة الدالة (ISCOLOR()) صحيحة أى (True) يتم التحويل إلى الألوان والعكس صحيح.

٢٦ - ٢ - ٦ تعديل وحدة الأقراص المستخدمة

يمكن لمخطط البرامج تعديل وحدة الأقراص المستخدمة فى أى مكان فى البرنامج حتى يمكن تحميل الملفات الموجودة فى قرص معين وذلك باستخدام الأمر (SET DEFAULT TO) ثم كتابة رمز وحدة الأقراص الموجود بها الملفات المراد تحميلها.

فمثلا لاستخدام القرص الصلب (C) يتم كتابة الأمر التالى :

SET DEFAULT TO C

وفى حالة وجود الملفات فى دليل فرعى (Subdirectory) داخل القرص الصلب مثلا يتم استخدام الأمر (SET PATH TO) ثم كتابة المسار المطلوب. فمثلا إذا كانت الملفات موجودة فى الدليل الفرعى (C:\cadets) يتم استخدام الأمر التالى :

SET PATH TO C:\Cadets

كما يمكن إلغاء المسار الذى سبق تحديده باستخدام الأمر التالى :

SET PATH TO

دون تحديد مسار معين.

٢٦ - ٢ - ٧ إعادة تعريف مفاتيح الوظائف (Function keys)

يسمح البرنامج لمخطط البرامج بإعادة تعريف تسع مفاتيح من مفاتيح الوظائف العشرة حيث أن مفتاح (F1) يكون محجوزا لشاشات المساعدة (Help) التى يستخدمها برنامج (+ DBase III). وحتى يتم استخدام مفتاح من مفاتيح الوظائف فى تنفيذ أمر معين يتم كتابة هذا الأمر بين علامات تنصيص (Quotation) مع كتابة الفاصلة المنقوطة فى نهاية الأمر (حيث أن الفاصلة المنقوطة (;) تمثل مفتاح الإدخال). فمثلا لتخصيص المفتاح (F2) للخروج من البرنامج ، يستخدم الأمر التالى :

SET FUNCTION 2 TO 'QUIT;'

ويمكن كتابة أى أمر بحيث لايزيد طوله عن ٣٠ حرفا متضمنا الفاصلة المنقوطة.

ملاحظة

يراعى قبل إنتهاء كتابة البرنامج إعادة مفاتيح الوظائف إلى حالتها الأولى. فمثلا لإعادة المفتاح (F2) إلى وظيفته الأولى يستخدم الأمر التالى :

SET FUNCTION 2 TO 'ASSIST; '

٢٦ - ٢ - ٨ التحكم فى عناوين الحقول (Headings)

عند استخدام الأمر (LIST) أو الأمر (DISPLAY) تظهر عناوين الحقول فى السطر الأول ويليها البيانات الخاصة بالسجلات المختلفة. وظهور هذه العناوين هو الوضع المبدئى (Default). وفى معظم الأحيان يحتاج مخطط البرامج إلى وضع عناوين مختلفة لهذه الحقول أو إظهار هذه العناوين بأشكال مختلفة عن الوضع المبدئى ولتنفيذ ذلك يتم كتابة الأمر التالى :

SET HEADING OFF

وهذا يؤدى إلى عدم ظهور عناوين الحقول ويمكن لمخطط البرامج بعد ذلك كتابة العناوين التى يريدها وبالطريقة التى يريدها كما سيتم الإيضاح فيما بعد.

٢٦ - ٢ - ٩ إخفاء رسالة المساعدة (Help Message)

عند كتابة أى أمر خطأ من مشيرة النقطة تظهر الرسالة التالية :

Do you want some help (y/n)?

فإذا أراد مخطط البرامج عدم ظهور هذه الرسالة أثناء تنفيذ البرنامج فإنه يستخدم الأمر التالى :

SET HELP OFF

ويمكن لمخطط البرامج بعد ذلك استخدام رسائل المساعدة التى يريدها كما يمكنه عرض شاشات مساعدة خاصة بالبرنامج توضح للمستخدم مكان الخطأ وطريقة تصحيحه. كما يمكن إخفاء المستطيل الذى يظهر أعلى الشاشة لتوضيح مفاتيح التصحيح أثناء الكتابة وذلك باستخدام الأمر :

SET MENU OFF

كما يمكن بعد ذلك تصميم شاشات التصحيح المناسبة للبرنامج.

٢٦ - ٢ - ١٠ إلغاء رسالة الأمان (Safety)

عندما يقوم البرنامج بنسخ ملف مكان ملف آخر أو تخزين ملف بعد تعديله تظهر رسالة للمستخدم لتحذيره والتأكد أنه يريد فعلا تنفيذ ذلك. وعادة لايريد مخطط البرامج ظهور هذه الرسائل للمستخدم أثناء تنفيذ البرنامج. ويتم تحقيق ذلك باستخدام الأمر (SET SAFETY OFF).

٢٦ - ٢ - ١١ إخفاء عمود الحالة (Status Bar)

فى معظم الأحيان لايريد مخطط البرامج ظهور عمود الحالة أسفل الشاشة. ولتنفيذ ذلك يتم كتابة الأمر (SET STATUS OFF) داخل البرنامج الرئيسى. كما يمكن أيضا إخفاء الرسالة التى تظهر أسفل عمود الحالة ووضع أى رسائل أخرى يريدها مخطط البرامج فى نفس المكان عن طريق استخدام الأمر :

SET MESSAGE TO

ثم كتابة الرسالة المطلوب ظهورها.

٢٦ - ٢ - ١٢ إخفاء لوحة الأهداف (Scoreboard)

عند إخفاء عمود الحالة (Status Bar) فإن برنامج (+ DBase III) يستخدم

أول سطر أعلى الشاشة فى إظهار بعض البيانات التى كانت تظهر على عمود الحالة ويسمى هذا السطر لوحة الأهداف (Scoreboard).

وقد يحتاج مخطط البرامج إلى عرض رسائل معينة على هذا السطر كما قد يحتاج إلى استغلال الشاشة كلها فى تصميم شاشة إدخال بيانات. فى هذه الحالة فإنه لايريد ظهور أى رسائل فجائية على هذا السطر. ويتم ذلك باستخدام الأمر التالى :

SET SCOREBOARD OFF

وهناك العديد من أوامر التجهيز الأخرى التى تكتب فى البرنامج الرئيسى والتى يستخدم فيها الأمر (SET) وسيتم دراستها بالتفصيل فى الجزء الخاص بالأوامر (Commands) فى الجزء الثالث من هذا الكتاب.

ملاحظة

عادة يتم كتابة مجموعة أوامر التجهيز التى سبق شرحها فى بداية أى برنامج رئيسى (Main Program). ويمكن توفيرا للجهد كتابة هذه الأوامر فى ملف أوامر (Command File) منفصل ونسخها فى أى برنامج جديد يراد تصميمه.

ملاحظة

ماسبق ذكره فى هذا الفصل ينطبق أيضا على كل برامج عائلة (DBase) مثل (DBase IV) ، (FoxBase) ، (FoxBase +) ، (FoxPro).



الفصل السابع والعشرون

التحكم فى الشاشة من خلال البرنامج



يعتبر تصميم شاشات إدخال البيانات (Custom Screens) من أهم وسائل تحقيق التفاعل بين البرنامج والمستخدم. فكلما كانت هذه الشاشات واضحة للمستخدم وقريبة إلى نماذج البيانات التى يستخدمها كلما كان من السهل عليه تشغيل البرنامج والإستفادة منه. كما أن عرض الرسائل والإرشادات الواضحة للمستخدم على الشاشة بطريقة واضحة ومفهومة يؤدى إلى سهولة متابعة البرنامج والحصول على أكبر كفاءة له.

والبرنامج يتيح لمخطط البرامج مرونة كبيرة فى تصميم شاشات إدخال البيانات وعرض أى رسائل للمستخدم بما يحقق التفاعل بينه وبين البرنامج.

٢٧ - ١ إحداثيات الشاشة

تنقسم شاشة الحاسب إلى ٢٥ سطرا أفقيا و٨٠ عمودا رأسيا ومن تقاطع هذه الصفوف مع الأعمدة تتحدد النقط التى يمكن كتابة حروف فيها. ويتم ترقيم الصفوف من أعلى إلى أسفل بدءا من الصفر (0) وإنتهاء بأربعة وعشرين (24). أما الأعمدة فترقم من اليسار إلى اليمين بدءا من الصفر (0) وانتهاء بتسعة وسبعين (79). ويتم تحديد موقع أى نقطة على الشاشة بكتابة رقم الصف (Row) أولا ثم رقم العمود (Column). فمثلا النقطة (1,5) هى النقطة الناتجة عن تقاطع الصف رقم واحد مع العمود رقم ٥ وهكذا.

ملاحظة

عند استخدام الأمر (SET STATUS ON) يصبح السطر رقم صفر أعلى الشاشة هو لوحة الأهداف (Scoreboard). فعندما يراد استخدام السطر الأول فى عرض أى بيانات على الشاشة من خلال البرنامج مع تلافى أى مشاكل تنتج عن عرض رسائل فجائية للبرنامج فى هذا السطر يتم استخدام الأمر (SET SCOREBOARD OFF).

٢٧ - ٢ إستخدام الأمر (@ ... SAY)

يستخدم هذا الأمر فى تصميم شاشات إدخال البيانات كما يستخدم فى عرض أى بيانات أو رسائل على الشاشة. فمثلا لعرض محتويات حقل الإسم (Name) لسجل معين بدءا من السطر الرابع والعمود التاسع (4 ,9) يتم استخدام الأمر التالى :

```
@ 4,9  SAY Name
```

وحرف (@) هنا يستخدم بمعنى عند أو "AT" وذلك لتحديد موقع بدء الكتابة. وعندما يراد عرض بيانات متغير ذاكرة (Memory Variable) يجب التأكد أولا من إنشاء هذا المتغير وذلك كالآتى مثلا :

```
STORE "Enter New Name" To message
@ 10,15 SAY message
```

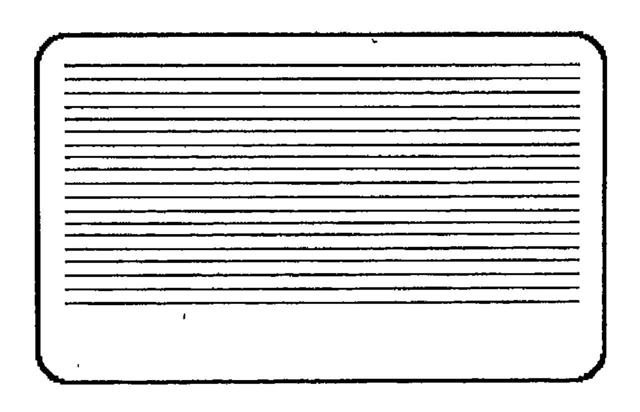
٢٧ - ٣ مسح الشاشة

عند عرض بيانات على الشاشة يجب أولا مسح البيانات السابقة الموجودة على الشاشة ويتم ذلك باستخدام الأمر (CLEAR). ويمكن مسح جزء فقط من الشاشة باستخدام الأمر (CLEAR ... @) مع كتابة الإحداثيات المطلوب بدء المسح عندها بعد الحرف (@).

فمثلا عندما يراد مسح الشاشة ابتداء من السطر ١٩ إلى آخر الصفحة مع ترك السطور من صفر إلى ١٨ على الشاشة يتم استخدام الأمر التالى :

```
@ 19,0 CLEAR
```

انظر الشكل (٢٧ - ١)



شكل (٢٧ - ١)

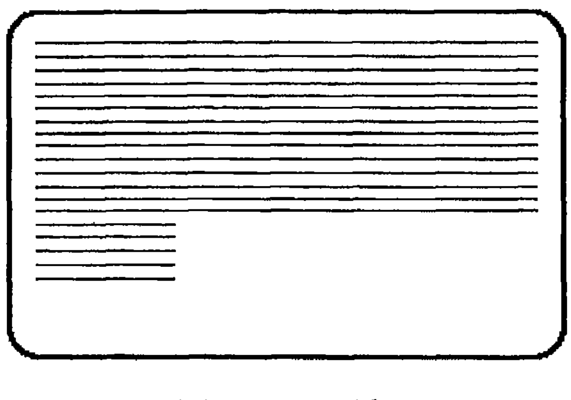
ملاحظة

يجب ملاحظة الفرق بين الأمر (CLEAR) المستخدم فى مسح الشاشة والأوامر (CLEAR ALL) أو (CLEAR MEMORY) المستخدمة فى مسح متغيرات الذاكرة.

كما يمكن مسح الشاشة إبتداء من سطر معين وعمود معين كالآتى مثلا :

@ 15,15 CLEAR

انظر الشكل (٢٧ - ٢)



شكل (٢٧ - ٢)

كما يمكن مسح سطر واحد عن طريق كتابة إحداثيى هذا السطر مع عدم كتابة أى شىء فيه. وذلك كالآتى مثلا :

@ 15,0

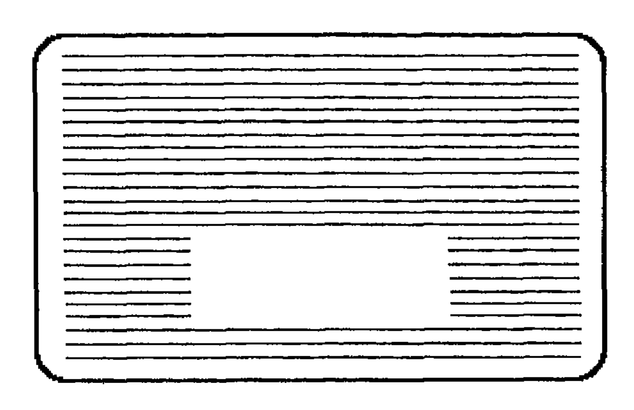
كما يمكن مسح مجموعة من السطور مسحا تدريجيا أى سطرا سطرا كالآتى مثلا :

```
@   19,0
@   20,0
@   21,0
@   22,0
@   23,0
@   24,0
```

كما يمكن مسح مساحة مستطيلة أو مربعة من الشاشة عن طريق تحديد إحداثى النقطة أعلى يسار هذا المستطيل والنقطة أسفل يمين هذا المستطيل. وذلك كالآتى مثلا :

```
@ 15,15 CLEAR TO 21,50
```

انظر الشكل (٢٧ - ٣)

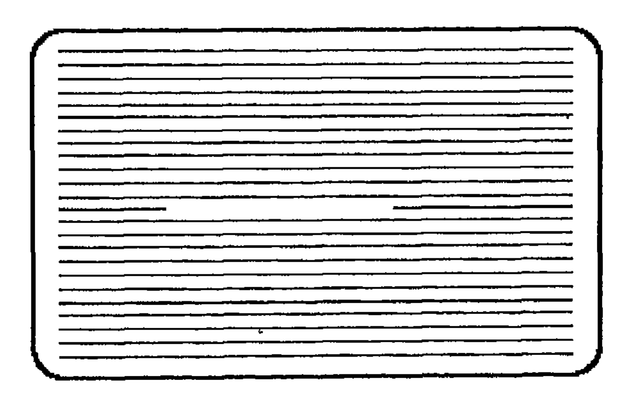


شكل (٢٧ - ٣)

كما يمكن مسح جزء من السطر فقط عن طريق كتابة مسافات خالية (Spaces) فى هذا السطر وذلك كالآتى مثلا :

```
@ 13,15 SAY SPACE ( 20 )
```

انظر الشكل (٢٧ - ٤)



شكل (٢٧ - ٤)

مما سبق يلاحظ أن البرنامج يتيح لمخطط البرامج أكبر مرونة ممكنة فى التحكم فى الشاشة وعرض ومسح أى رسائل أو بيانات فى أى مكان منها.

٢٧ - ٤ عرض نص على الشاشة

كما سبق الإيضاح فإن الأمر (SAY ... @) يستخدم فى عرض أى رسائل (Messages) أو بيانات على الشاشة. وعندما يراد عرض نص كبير (Text) على الشاشة يمكن استخدام الأمر (SAY ... @) أيضا فى كتابة كل سطر فى النص على سطر محدد فى الشاشة. ولكن هذه الطريقة قد تبدو طويلة وتستهلك كثيرا من الوقت خصوصا فى النصوص الكبيرة التى يراد عرضها من خلال البرنامج مثل شاشات المساعدة (Help Screens). لذلك فالبرنامج يتيح وسيلة أخرى أسرع فى كتابة النصوص الكبيرة وذلك باستخدام الأمر (TEXT....ENDTEXT). حيث يتم كتابة (TEXT) فى أول سطر قبل النص المراد كتابته ثم كتابة النص المطلوب ثم كتابة (ENDTEXT) بعد آخر سطر فى النص.

ملاحظة

يجب ملاحظة أن الأمر (TEXT ... ENDTEXT) لايستخدم السطر رقم صفر فى كتابة النص كما أنه لايستخدم فى عرض بيانات الحقول أو متغيرات الذاكرة.

٢٧ - ٥ إستخدام الأمر (READ ... GET ... @)

يستخدم هذا الأمر فى إستقبال المدخلات التى يقوم المستخدم بإدخالها. والجزء الأول من الأمر وهو (GET ... @) يؤدى إلى عرض عمود ضوئى (Highlight) بدءا من النقطة التى تكتب احداثياتها بعد الحرف (@). كما يستخدم الجزء الآخر (READ) فى تخزين البيانات التى يكتبها المستخدم فى هذا العمود الضوئى فى المتغير الذى يتم كتابة إسمه بعد الأمر (GET). وذلك كالآتى مثلا :

```
@ 15,15 GET mname READ
```

ويجب التأكد أن متغير الذاكرة (mname) قد سبق إنشاؤه.

ويستخدم هذا الأمر فى تصميم شاشات إدخال بيانات مثل الشاشات التى تظهر عند استخدام الأمر(APPEND) أو الأمر (EDIT). حيث يتم كتابة هذا الأمر مع متغيرات الذاكرة التى يتم إنشاؤها ممثلة لحقول البيانات فى ملف قاعدة البيانات على أن يتم تحديد مكان العمود الضوئى (Highlight) الممثل لهذا الحقل عن طريق الاحداثيات التى تكتب بعد الحرف (@). كما يتم كتابة رسائل للمستخدم قبل العمود الضوئى لتوضيح البيان المطلوب إدخاله وذلك باستخدام الأمر (SAY ... @).

فمثلا إذا كانت هناك ثلاثة حقول فى ملف قاعدة البيانات للإسم والعنوان ورقم التليفون وهى (name ، address ، tel) على الترتيب فى هذه الحالة يجب أولا إنشاء متغيرات الذاكرة (mname ، maddress ، mtel) على الترتيب مثلا ثم كتابة الأوامر التى تؤدى إلى ظهور الشاشة المطلوبة وذلك كالآتى :

```
mname    = SPACE(20)
maddress = SPACE(20)
mtel   = SPACE(10)
CLEAR
@  3,2  SAY 'Enter Name:'
@  3,30 GET mname
@  5,2  SAY 'Enter Address:'
@  5,30 GET 'maddress'
@  7,2  SAY 'Enter Telephone No:'
@  7,30 GET mtel
READ
```

وعند تنفيذ البرنامج تظهر الشاشة المبينة فى الشكل (٢٧ - ٥).

ويلاحظ فى هذا المثال استخدام أمر (READ) واحد لجميع السطور. وهذا يؤدى إلى ظهور جميع الأعمدة الضوئية (Highlights) مرة واحدة على الشاشة. كما يؤدى إلى ظهور مؤشر صغير فى آخر عمود ضوئى (Highlight) يساعد على تصحيح أى بيانات يقوم المستخدم بإدخالها خطأ. كما يمكن نقل المؤشر إلى أى عمود ضوئى آخر وتصحيح البيانات التى سبق إدخالها. وذلك باستخدام مفاتيح الأسهم (--> , <-- , ↓ , ↑).

Enter Name :
Enter Address :
Enter Telephone No:

شكل (٢٧ - ٥)

وهناك طريقة أخرى لاستخدام الأمر (READ) وذلك بكتابته بعد كل أمر (GET). وهذا يؤدى إلى ظهور أول عمود ضوئى فقط والإنتظار حتى يكتب المستخدم بيانات فى هذا العمود ثم يضغط على مفتاح الإدخال فيظهر العمود التالى وينتظر حتى يكتب المستخدم فيه ويضغط على مفتاح الإدخال فيظهر العمود الثالث ... وهكذا. ويمكن استخدام هذه الطريقة مع نفس المثال السابق كالآتى :

```
CLEAR
@   3,2  SAY 'Enter Name:'
@   3,30 GET mname
READ
@   5,2  SAY 'Enter Address:'
@   5,30 GET 'maddress'
READ
@   7,2  SAY 'Enter Telephone No:'
@   7,30 GET mtel
READ
```

وفى هذه الحالة لايستطيع المستخدم الرجوع إلى الحقول السابقة لإصلاح البيانات التى قام بإدخالها فيها إلا باستخدام الأمر (READ SAVE) بدلا من الأمر (READ). حيث أن الأمر (READ SAVE) يمكن المستخدم من الرجوع إلى الحقول السابقة عن طريق مفاتيح الأسهم (--> , <-- , ↓ , ↑) وتصحيح البيانات الموجودة فيها.

ويمكن عرض عدة أعمدة ضوئية (Highlights) على سطر واحد وذلك عن طريق تثبيت رقم السطر وتغيير رقم العمود. ويمكن ملاحظة ذلك فى السطور التالية :

```
@ 5,0 GET mname
@ 5,30 GET maddress
@ 5,60 get mtel
READ
```

وعند تنفيذ هذه السطور يظهر الآتى على الشاشة :

ملاحظة

يجب ملاحظة أن عدد أماكن التخزين المسموح بها للأمر (GET) لايزيد عن ١٢٨ مكانا وهو يعتبر عددا كافيا جدا لأن كل مكان من هذه الأماكن يتم مسحه آليا بمجرد استخدام الأمر (READ) فى تخزين البيانات الموجودة فيه. ولكن هذا المكان يظل محجوزا فى حالة استخدام الأمر (READ SAVE) وفى هذه الحالة يلزم استخدام الأمر (CLEAR GETS) لمسح الأماكن المحجوزة والإحتفاظ بعدد كبير من الـ (GETS) التى يمكن استخدامها فى إدخال مدخلات جديدة.

٢٧ - ٦ إنشاء شاشة مكونة من عدة صفحات

عندما يكون عدد حقول قاعدة البيانات كبيرا ولايمكن عرضه على شاشة واحدة فإن البرنامج يتيح لمخطط البرامج تصميم شاشة إدخال مكونة من عدة صفحات وذلك باستخدام الأمر (CLEAR) لمسح كل شاشة والإنتقال إلى الشاشة التالية. ويمكن فى هذه الحالة استخدام أمر (READ) مرة واحدة بعد أوامر (GET) مع ملاحظة استخدام الأمر (GLEAR GETS) عندما يزيد عدد الـ (GETS) عن ١٢٨.

ملاحظة

عند إدخال البيانات فى العمود الضوئى (Highlight) ينتقل المؤشر إلى العمود الضوئى التالى فى حالة إمتلاء العمود الضوئى الأول. أما فى حالة عدم إمتلاء العمود

الأول بالبيانات فيجب فى هذه الحالة الضغط على مفتاح الإدخال حتى ينتقل المؤشر إلى العمود التالى. وهذا هو نفس ما يحدث عند استخدام الأمر (APPEND) أو الأمر (EDIT).

ويمكن تغيير هذا الوضع - أى عدم انتقال المؤشر إلى العمود التالى إلا بالضغط على مفتاح الإدخال بصرف النظر عن امتلاء العمود الضوئى أو عدم امتلائه - وذلك باستخدام الأمر (SET CONFIRM ON).

٢٧ - ٧ استخدام الأمر (ACCEPT) والأمر (INPUT)

كما سبق الإيضاح فإن الأمر (GET...@) يستخدم عادة عندما يراد استقبال المدخلات من المستخدم فى عدة متغيرات ذاكرة. أما اذا أريد الإدخال فى متغير ذاكرة واحد فقط مثل الإجابة على سؤال معين وتخزين هذه الاجابة فى متغير ذاكرة فى هذه الحالة يفضل استخدام الأمر (ACCEPT) أو الأمر (INPUT) حيث أن استخدامهما يكون أسرع وأسهل.

والأمر (ACCEPT) والأمر (INPUT) متشابهان فى الوظيفة فهما يطلبان من المستخدم كتابة شىء معين ثم يخزنان ما يكتبه المستخدم فى متغير ذاكرة. هذا المتغير يمكنه تخزين حتى ٢٥٤ حرفا. وهما يختلفان عن الأمر (GET ... @) فى أنهما لايتطلبان إنشاء متغير الذاكرة أولا لأنهما يقومان بهذه العملية آليا كما أنهما لايعرضان عمودا ضوئيا (Highlight) للكتابة فيه.

والفرق الوحيد بين الأمرين (ACCEPT) و (INPUT) هو أن الأمر (ACCEPT) يقبل مدخلات حرفية (Character) فقط. حتى إذا أدخل المستخدم أعدادا فإنه يعامل هذه الأعداد كحروف. أما الأمر (INPUT) فإنه يقبل أى نوع من المدخلات سواء كانت حرفية (Character) أو عددية (Numeric) أو تاريخية (Date) أو منطقية (Logical) ثم يقوم بإنشاء متغير الذاكرة المقابل. وعند إدخال مدخلات حرفية يشترط وضعها بين علامات تنصيص (Quotation Marks) وهذا عكس الأمر (ACCEPT) الذى لايتطلب وضع علامات التنصيص.

وعند كتابة أى أمر من هذين الأمرين يتم كتابة رسالة للمستخدم أولا ثم تحديد إسم متغير الذاكرة الذى يتم إدخال ما يكتبه المستخدم فيه. وذلك كالآتى مثلا :

```
ACCEPT "What is your name?" TO mname
```

وهذا الأمر يطلب من المستخدم كتابة إسمه ثم يخزن هذا الإسم فى متغير الذاكرة (mname).

كما يمكن كتابة الأمر التالى :

INPUT "Enter your age" TO mage

وهذا الأمر يطلب من المستخدم كتابة قيمة عددية تمثل عمره ثم يخزن هذه القيمة العددية فى المتغير (mage).

٢٧ - ٨ إستخدام الأمر (WAIT)

يستخدم الأمر (WAIT) لعمل توقف مؤقت للبرنامج (Pause) حتى يقوم المستخدم بالضغط على أى مفتاح لاستكمال تنفيذ البرنامج وذلك حتى يعطى المستخدم الفرصة لقراءة رسائل معينة أو بيانات على الشاشة. ويفضل استخدام الأمر (SET ESCAPE OFF) كما سبق الإيضاح حتى لايتسبب ضغط المستخدم على مفتاح الهروب (Esc) فى توقف البرنامج تماما.

والأمر (WAIT) يؤدى إلى عرض الرسالة التالية :

Press any key to continue

وذلك عند كتابته دون كتابة أى رسالة معه كالآتى مثلا :

WAIT

ويمكن لمخطط البرامج كتابة أى رسالة يريد عرضها للمستخدم. وذلك كالآتى مثلا :

WAIT "Press any key to return to main menu"

وهناك فرق واضح بين الأمر (WAIT) والأمرين (ACCEPT) ، (INPUT) حيث أن الأمر (WAIT) لايخزن ما يكتبه المستخدم فى متغير ذاكرة (Memory Variable). وانما يقوم بإيقاف تنفيذ البرنامج إيقافا مؤقتا حتى يضغط المستخدم على أى مفتاح للإستمرار. ولكن مع ذلك فيمكن فى بعض الحالات تخزين الحرف

الذى يقوم المستخدم بإدخاله فى متغير ذاكرة عندما يتطلب البرنامج ذلك. ويمكن ملاحظة ذلك من مجموعة الاوامر التالية :

```
WAIT "press R to return to mainmenu , or " + ;
" any other  key to continue " TO mmenu
IF UPPER (mmenu) = "R"
    RETURN
ELSE
    -------
    -------
    -------  commands
    -------
    -------
ENDIF
```

هذه الاوامر تؤدى إلى توقف تنفيذ البرنامج حتى يضغط المستخدم على أى حرف. فإذا ضغط على الحرف (R) يتم الرجوع إلى القائمة الرئيسية وإذا ضغط على أى مفتاح آخر يستمر تنفيذ أوامر البرنامج. ويلاحظ هنا استخدام علامة (;) لإمتلاء السطر الاول وإكمال كتابة الأمر فى السطر التالى.

ملاحظة

لايمكن استخدام إحداثيات الشاشة فى تحديد مكان ظهور الرسالة الخاصة بالأوامر الثلاثة (ACCEPT) ، (INPUT) ، (WAIT) ولكن يمكن التحكم فى أماكن هذه الرسائل عن طريق إضافة سطور خالية بعد آخر رسالة معروضة على الشاشة. ويتم ذلك باستخدام الامر (?) كما سيتم الإيضاح فيما بعد.

ملاحظة

ماسبق ذكره فى هذا الفصل ينطبق أيضا على كل برامج عائلة (DBase) مثل (DBase IV) ، (FoxBase) ، (FoxBase +) ، (FoxPro).



الفصل الثامن والعشرون

التحكم فى شكل ومدى المدخلات

(Templates and Ranges)



من المهم جدا أن يحتوى تصميم البرنامج على إمكانيات التحكم فى شكل المدخلات (Template) ومدى هذه المدخلات (Range) لأن هذا التحكم يؤدى إلى تأمين قاعدة البيانات ضد أى مدخلات خطأ يقوم المستخدم بإدخالها. فمثلا عندما يكون مطلوبا إدخال قيمة عددية فى حقل معين فإن تحديد الشكل (Template) يؤدى إلى قبول مدخلات عددية ولا يقبل أى مدخلات أخرى. كما أن تحديد مدى معين (Range) لهذه الأعداد يؤدى إلى تأمين البرنامج ضد إدخال أى قيم قد تؤدى إلى توقف البرنامج.

ولتحديد شكل المدخلات يستخدم التعبير (PICTURE) ولتحديد مدى المدخلات يستخدم التعبير (RANGE). وسيتم دراسة هذين التعبيرين بالتفصيل فى الأجزاء التالية.

٢٨ - ١ إستخدام التعبير (PICTURE)

يستخدم التعبير(PICTURE) كمكمل للأمر(SAY...@) والأمر(Get...@). ويلى هذا التعبير خليط من واحد أو أكثر من الرموز الممثلة للشكل (Template Symbols) والدوال (Functions).

وعند استخدام هذا التعبير (PICTURE) مع الأمر (Say...@) فإنه لايغير البيانات المخزنة فى الحقول أو متغيرات الذاكرة ولكنه يغير شكل هذه البيانات عند ظهورها على الشاشة. أما استخدامه مع الأمر (GET...@) فإنه يؤدى إلى التحكم فى البيانات التى يقوم المستخدم بإدخالها قبل تخزينها فى الحقول أو متغيرات الذاكرة.

٢٨ - ٢ إستخدام رموز الشكل (Template Symbols)

تستخدم رموز الشكل فى تحديد نوع الحروف التى يتم إدخالها. كما يمثل عدد رموز الشكل عدد الحروف التى يتم كتابتها فمثلا يمكن ملاحظة المثال التالى :

```
@ 15,15 SAY mprice Picture "999999.99 "
```

والرقم (9) هنا يمثل رمز يوجه البرنامج إلى إظهار أعداد فقط كما يمثل عدد أرقام (9) عدد الأرقام التى يسمح البرنامج بظهورها. كما تؤدى النقطة (.) إلى تحديد عدد الكسور العشرية.

والوضع المبدئى لظهور الأعداد عند استخدام الامر (GET... @) هو ظهور عشرة أرقام. فإذا أريد عرض أو إدخال أعداد أكبر من ذلك يتم كتابة عدد من الرموز يمثل هذه الأرقام. فمثلا يمكن إدخال الأوامر التالية :

```
mamount = 0
@ 15,15 GET mamount PICTURE"999999999999"
```

وهذه الأوامر تؤدى إلى الحصول على عدد مكون من ١٢ رقما.

ويلاحظ هنا ضرورة إنشاء متغير الذاكرة (mamount) قبل استخدام الأمر (GET...@) كما سبق الإيضاح. وفى هذه الحالة يتم إعطاؤه القيمة صفر حيث أن أى قيمة عددية تكون كافية لإنشاء المتغير.

ومن الرموز التى تستخدم فى أغلب البرامج الرمز (!) وهذا يؤدى إلى تحويل الحروف التى يدخلها المستخدم إلى حروف كبيرة(Uppercase). فمثلا يمكن إدخال الأوامر التالية :

```
mchoice = " "
@ 10,5 SAY "Enter your choice A,B,C,D"
@ 10,50 GET mchoice PICTURE "!"
READ
```

وتؤدى هذه الأوامر إلى ظهور رسالة للمستخدم تطلب منه إدخال حرف معين من الحروف (A,B,C,D). وعند كتابة المستخدم لهذا الحرف سواء كتبه صغيرا أو كبيرا يتم تحويله إلى حرف كبير (Uppercase) ثم تخزينه فى المتغير (mchoice).

ويلاحظ هنا أنه تم كتابة رمز واحد لأن المتغير يحتوى على حرف واحد فقط. أما إذا أريد إدخال عدد معين من الحروف فيتم كتابة عدد من الرموز يماثل عدد الحروف المطلوب إدخالها. فمثلا يمكن كتابة الأمر التالى :

```
@ 15,15 GET mname PICTURE "AAAAAA "
```

ويؤدى هذا الأمر إلى إدخال إسم يحتوى على ستة حروف والحرف (A) هنا يعنى قبول الحروف الأبجدية فقط ولايقبل الأعداد أو الحروف الخاصة (Special characters).

وعندما يراد إدخال بيانات حرفية تحتوى على أعداد أو حروف خاصة يتم استخدام الرمز (X) و الأمر التالى يوضح ذلك :

```
@ 10,10 GET maddr PICTURE "XXXXXXXXXXXXXXXXXXXX "
```

هذا الأمر يعنى قبول ٢٠ حرفا تمثل العنوان (maddr) وتحتوى على أى نوع من الحروف سواء كانت أرقاما أو حروفا خاصة.

ويمكن التحكم فى شكل الأعداد التى تظهر على الشاشة باستخدام عدة رموز من رموز الشكل (Template Symbols). فمثلا علامة ($) التى تسمى علامة الدولار (Dollar Sign) يتم كتابتها بعد التعبير (PICTURE) وهى تؤدى إلى ظهور هذه العلامة مكان أى أصفار يسار العدد كما يتضح من الأمر التالى :

```
@ 10,10 SAY mamount PICTURE  "$$$$$$.$$"
```

فإذا كان المتغير (mamount) يحتوى على العدد (66) مثلا يظهر العدد على الشاشة كالآتى :

```
$$$$66.00
```

وإذا أريد إظهار علامة دولار واحدة قبل العدد يتم استخدام الدالة (STR) التى سيتم شرحها فيما بعد وذلك يتضح من الأمر التالى :

```
@ 10,10 SAY "$" + STR (mamount, 5,2)
```

ويؤدى هذا الأمر إلى إظهار العدد على الصورة التالية :

```
$ 66.00
```

ويلاحظ هنا استخدام علامة الجمع (+) فى جمع البيانات الحرفية (Concatination). وهذا سيتم شرحه فيما بعد.

كما يمكن استخدام الفاصلة(Comma) داخل العدد كما يتضح من السطور التالية :

```
mprice = 0.00
@ 10,10 GET mprice PICTURE "999,999.99"
READ
```

ومن الرموز التى تستخدم فى المدخلات المنطقية الرمز (Y) وهو يحدد الحرف المنطقى الذى يدخله المستخدم ليكون (Y) وتعنى نعم (Yes) أو يكون (N) وتعنى لا (N0). ويتضح ذلك من السطور التالية :

```
@ 10,10 SAY "Send reports to printer (Y/N) " ;
GET mprint PICTURE "Y"
READ
```

٢٨ - ٣ إستخدام دوال الشكل (Template Functions)

دوال الشكل هى دوال تتحكم فى شكل المدخلات كلها وليس فى حرف واحد أى تستخدم دالة واحدة فقط فى التحكم في شكل الحروف كلها. ويمكن استخدام الدوال مع رموز الشكل فى أمر واحد. ولكن يجب أن تكون الدالة فى أول العبارة التى تلى التعبير (PICTURE) ويتم فصلها عن الرموز بمسافة واحدة (Space). وتبدأ الدالة عادة بالحرف(@) ثم الحرف الذى يمثل هذه الدالة. فمثلا الأمر التالى يوضح إحدى هذه الدوال.

```
@ 5,5 SAY mcost PICTURE "@B 9,999,999/99 "
```

وهذا الأمر يؤدى إلى عرض الرقم مضبوطا من اليسار (Left Justified) بدلا من عرضه مضبوطا من اليمين كما يحدث فى العادة. كما يؤدى هذا الأمر أيضا إلى عرض الأعداد وبها فواصل (Commas) بعد كل ثلاثة أرقام.

ويمكن استخدام كلمة (Function) بدلا من الحرف (@) ولكنها فى هذه الحالة تكتب قبل التعبير (PICTURE) كما فى الأمر التالى :

```
@ 5,5 SAY mcost FUNCTION "B" PICTURE "9,999,999.99 "
```

كما يمكن جمع عدة دوال فى أمر واحد فمثلا فى برامج المحاسبة عندما يراد عرض نوع الرصيد إذا كان دائنا (Credit) ومدينا (Debit) بدلا من علامة السالب والموجب على الترتيب يستخدم الأمر التالى :

```
@ 5,5 SAY mamount PICTURE "@XC 999,99 "
```

حيث تؤدى الدالة (X) إلى إظهار الحروف (DB) أى مدين (Debit) بعد العدد السالب. كما تؤدى الدالة (C) إلى إظهار الحروف (CR) أى دائن (Credit) بعد العدد الموجب.

ويمكن استخدام الدالة (Z) التى تمسح الأصفار الموجودة يسار الرقم أو أى أصفار أخرى ليس لها قيمة.

كما تستخدم الدالة (E) فى تحويل التاريخ (Date) إلى الشكل الأوربى (dd/mm/yy). والدالة (D) فى تحويل التاريخ إلى الشكل الأمريكى (mm/dd/yy). والأوامر التالية توضح ذلك :

```
mdate = CTOD ('03/10/89')
@ 5,5 SAY mdate PICTURE '@E'
```

وهذه الأوامر تؤدى إلى عرض التاريخ (mdate) على الصورة (10/03/89).

ويلاحظ من الأمر فى السطر الأول إستخدام الدالة (CTOD) وهى إختصار (Character To Date Conversion Function) لتحويل التاريخ من الحروف (Characters) إلى تاريخ (Date) حتى يتم تخزينه فى المتغير التاريخى (mdate).

كما تستخدم الدالة (R) فى إدخال أى حروف داخل العبارة الخاصة بالتعبير (PICTURE) بحيث تستخدم هذه الحروف كفواصل للحروف التى يدخلها المستخدم. فمثلا عندما يراد عرض الحروف وبينها مسافات يمكن كتابة الأمر التالى :

```
@ 5,5 SAY mname PICTURE "@R X X X X X X "
```

فى هذه الحالة تظل المسافات كما هى وتظهر الحروف مكان حروف (X). فإذا كان المتغير(mname) يحتوى على الإسم(AHMED) مثلا يظهر الإسم على الشاشة كالآتى :

```
A H M E D
```

ويجب ملاحظة أن هذه الحروف التى يتم إضافتها كفواصل لاتخزن بعد ذلك فى المتغير عند استخدام الأمر (GET) مثلا ، ولكنها تظهر على الشاشة فقط.

كما تستخدم الدالة (S) فى عمل إزاحة أفقية (Horizontal Scrolling) فى العمود الضوئى (Highlight) الذى يظهر مع الأمر (GET ... @) ويفيد هذا عندما يكون هناك مكان محدود للكتابة فيه ويراد كتابة بيانات تزيد عن العمود الضوئى. حيث تسمح هذه الإزاحة بإضافة حروف جديدة دون زيادة طول العمود الضوئى ويتم ذلك عن طريق كتابة رقم بعد الدالة (S) يحدد طول العمود الضوئى المراد عرضه. كما يتم إضافة رموز للشكل (Template Symbols) تمثل أقصى عدد من الحروف المطلوب إدخالها. ويتضح ذلك من مجموعة الأوامر التالية :

```
mname = SPACE(20)
@ 10,10 "Enter a name" ;
GET mname PICTURE "@ S8 !!!!!!!!!!!!!!!!!!!!"
READ
```

Ahmed Za

قبل الازاحة

med Zaky

بعد الازاحة

فعندما يتم كتابة إسم يزيد عن طول العمود الضوئى يلاحظ تحرك الحروف جهة اليسار حتى تسمح بإدخال الحروف الجديدة.

وهناك عدة دوال شكل (Template Functions) ورموز شكل (Template Symbols) يمكن استخدامها فى التحكم فى شكل المدخلات. ولكن لامجال لشرحها بالتفصيل فى هذا الجزء. إرجع الى الجزء الخاص بالدوال (Functions) فى الجزء الثالث من الكتاب.

٢٨ - ٤ تحديد المدى (RANGE)

يستخدم التعبير (RANGE) لتحديد المدى بالنسبة للمدخلات العددية أو التاريخية. حيث يستخدم مع الأمر (@...GET) لتحديد أصغر وأكبر قيمة يمكن للمستخدم إدخالها. ويجب ملاحظة أن التواريخ (Dates) هى فى الحقيقة أعداد حيث أن كل تاريخ يمثل داخل قاعدة البيانات بعدد معين.

فمثلا عندما يراد إدخال عدد فى المتغير (mprice) يكون محصورا بين عشرة وألف يتم كتابة الأمر التالى :

```
@ 5,5 GET mprice RANGE 10,1000
```

أما إذا أريد تحديد قيمة المتغير بحيث لاتقل عن (10) فى حين يمكن أن تزيد إلى أى عدد يتم كتابة الأمر التالى :

```
@ 5,5 GET mprice RANGE 10,
```

وبالنسبة للمتغيرات التاريخية يجب استخدام الدالة (CTOD) فى تحويل التاريخ من الحروف إلى التواريخ المقابلة. ويمكن ملاحظة ذلك من الأمر التالى :

@5,5 GET mdate RABGE CTOD('02/11/88),CTOD('08/08/89')

وعندما يقوم المستخدم بإدخال أى قيمة عددية أو تاريخية خارج المدى الذى تم تحديده يقوم برنامج (DBase III +) بعرض المدى المسموح باستخدامه على عمود الحالة (Status bar) أو على لوحة الأهداف (Scoreboard) كما يقوم بإرشاد المستخدم إلى الضغط على مسطرة المسافات (Space Bar) لمسح القيمة التى سبق إدخالها والمحاولة مرة ثانية.

٢٨ - ٥ إستخدام التعبير (TRANSFORM)

يستخدم التعبير (PICTURE) مع الأوامر (@ ... SAY) ، (@ ... GET) فقط ولكن لايمكن استخدامه مع باقى أوامر برنامج (DBase III +) مثل الأوامر :

? , ?? , DISPLAY , LABEL , LIST , REPORT

لذلك يستخدم التعبير (TRANSFORM) فى تحديد شكل المدخلات مع هذه الأوامر حيث يتم كتابة التعبير (TRANSFORM) يليه المتغير المراد تحديد شكله ثم الدالة والرموز المستخدمة فى تحديد الشكل المطلوب. والأمر التالى يوضح هذه العملية :

LIST TRANSFORM (mname , "@R X X X X X X X X ")

وهذا التعبير (TRANSFORM) يكون مفيدا جدا عند استخدام الأمر (CREATE REPORT) فى تصميم التقارير المطلوبة للبرنامج.

ملاحظة

ماسبق ذكره فى هذا الفصل ينطبق أيضا على كل برامج عائلة (DBase) مثل (DBase IV) ، (FoxBase) ، (FoxBase +) ، (FoxPro).

الفصل التاسع والعشرون

الدوال المستخدمة مع المدخلات



عند عرض البيانات على الشاشة يتطلب الأمر فى بعض الأحيان عرض بيانات حقول أو متغيرات ذاكرة مختلفة النوع. فمثلا يمكن عرض بيانات حقل حرفى مع حقل عددى أو تاريخى. وبرنامج (+ DBase III) لايسمح بكتابة أنواع مختلفة من الحقول أو متغيرات الذاكرة ذات الأنواع المختلفة على نفس السطر فى البرنامج الذى يتم كتابته. لذلك يلزم إجراء عمليات تحويل للبيانات الموجودة فى الحقول أو متغيرات الذاكرة. وتستخدم مجموعة من الدوال فى التحويل من نوع إلى آخر كما أن هناك دوالا تؤدى وظائف أخرى يتم إلقاء الضوء عليها فى الأجزاء التالية.

٢٩ - ١ الدوال الحرفية

٢٩ - ١ - ١ إستخدام الدالة (STR)

تستخدم هذه الدالة فى تحويل أى قيمة عددية إلى الحروف المقابلة (String) فمثلا عند كتابة الأمر التالى :

```
? STR ( 500 )
```

والضغط على مفتاح الإدخال يتم تحويل العدد(500) إلى الحروف المقابلة كالآتى:

```
"       500 "
```

ويلاحظ هنا وجود مسافات خالية قبل العدد وذلك لأن الدالة تعطى سلسلة حرفية (Character String) طولها عشرة حروف.

ويمكن التحكم فى طول السلسلة الحرفية وكذلك فى عدد الأرقام العشرية عن طريق إضافة عددين بعد العدد المطلوب تحويله العدد الأول يحدد طول السلسلة (String) والعدد الثانى يحدد عدد الأرقام العشرية (Decimal Numbers). فمثلا يمكن كتابة الأمر التالى :

```
? STR ( 11.14 , 5 , 2 )
```

ويلاحظ فى هذه الحالة ظهور العدد كالآتى :

11.14

ويجب ملاحظة أن الأمر (?) هو أمر من أوامر (+ DBase III) ويؤدى إلى إظهار نتيجة أى علاقة تلى الأمر وهو يعنى الإستفهام عن قيمة علاقة معينة.

ويلاحظ من المثال السابق أن طول سلسلة الحروف تم تحديده عن طريق الرقم (5) وذلك أخذا فى الإعتبار أن نقطة الكسر العشرى (Decimal Point) تحسب ضمن طول السلسلة. والأمر التالى يوضح استخدام الأمر (STR) بشكل آخر.

```
STORE 11.15 To X
STORE STR ( X*10,5 ) To Y
?
```

ويلاحظ عند كتابة الأمر الأخير والضغط على مفتاح الإدخال ظهور السلسلة الحرفية التالية :

111

ويلاحظ فى هذه الحالة إختفاء الكسر العشرى وذلك لأن الدالة لم تتضمن الرقم الذى يحدد عدد الأرقام العشرية.

أما إذا أريد إظهار الكسر العشرى كما هو فيتم كتابة الأمر التالى :

```
STR ( X*10 , 5 , 2 )
```

يلاحظ فى هذه الحالة ظهور العدد كالآتى :

111.5

وعند وجود أى تناقض فى المعاملات الخاصة بالدالة (STR) فإن البرنامج يعطى رسالة خطأ (Error message). فمثلا عند كتابة الأمر التالى :

```
? STR ( NUM , 1 , 2 )
```

والضغط على مفتاح الإدخال تظهر رسالة خطأ وذلك لأن عدد الأرقام العشرية أكبر من العدد الكلى للأرقام الموجودة فى العدد.

٢٩ - ١ - ٢ إستخدام الدالة (VAL)

تستخدم الدالة (VAL) فى تحويل البيانات الحرفية (Strings) إلى قيم عددية وهى عكس العملية التى تقوم بها الدالة (STR). فمثلا عند كتابة الأوامر التالية :

```
STORE ' 886.67 ' TO string
? VAL ( string )
```

مع الضغط على مفتاح الإدخال بعد كل أمر يلاحظ ظهور العدد التالى :

```
886.67
```

ومع أن العدد لم يتغير إلا أنه أصبح قيمة عددية يمكن إجراء أى عمليات حسابية عليها.

ملاحظة

عند استخدام الدالة (VAL) مع بيانات حرفية (Strings) لاتحتوى على أعداد فإن البرنامج يعطى القيمة صفر. فمثلا عند كتابة الأمر التالى :

```
?  VAL ( 'Hello' )
```

والضغط على مفتاح الإدخال يلاحظ ظهور القيمة صفر.

٢٩ - ١ - ٣ مقارنة البيانات الحرفية

رغم أن المقارنة دائما ترتبط فى الذهن بالقيم العددية حيث أنها القيم التى يسهل مقارنتها واستنتاج القيمة الأكبر أو القيمة الأصغر إلا أن الحروف والبيانات

الحرفية أيضا يمكن مقارنتها. وتستخدم هذه المقارنة فى بعض البرامج التى يتم كتابتها بواسطة برنامج (+ DBase III) أو برامج عائلة (DBase) الأخرى مثل (DBase IV) و (+ FoxBase) و (FoxPro). وتعتمد هذه المقارنة على أن كل حرف له كود الآسكى الخاص به. فعند مقارنة حرفين يتم مقارنة العدد الممثل لكود الآسكى (ASCII Code) فى كلا الحرفين. فمثلا عند كتابة الأمر التالى :

```
? 'A'  <  'a'
```

والضغط على مفتاح الإدخال تظهر القيمة (.T.) أى (True). وذلك لأن كود الآسكى الخاص بالحرف (A) وهو (65) أصغر من كود الآسكى الخاص بالحرف (a) وهو (97).

وبالمثل يمكن كتابة الأمر التالى :

```
? '950'  >  '750'
```

ويجب ملاحظة أن المقارنة هنا بين بيانات حرفية رغم أنها تحتوى على أعداد. فعند الضغط على مفتاح الإدخال تظهر نتيجة المقارنة وهى (.T.) أى (True) وذلك لأن كود الآسكى الخاص بالرقم ٩ وهو (57) أكبر من كود الآسكى الخاص بالرقم (7) وهو (55).

ويمكن مقارنة بيانات حرفية باستخدام معامل التساوى (=) مع ملاحظة أن المقارنة تتم حرفا حرفا حتى تنتهى السلسلة الحرفية (String) الموجودة يمين علامة التساوى. وفى هذه الحالة يعطى البرنامج القيمة (True) أى صحيح. فمثلا عند كتابة الأمر التالى :

```
? 'abcd' = 'abc'
```

والضغط على مفتاح الإدخال تظهر القيمة (.T.) أى (True).

أما عند كتابة الأمر التالى :

```
? ' abc ' = ' abcd '
```

والضغط على مفتاح الادخال فتظهر القيمة (.F.) أى (Fasle). وذلك لأن الحروف يسار علامة التساوى تنتهى قبل الحروف يمينها.

تحـذير

يجب ملاحظة عدم مقارنة قيم مختلفة فى النوع. فمثلا عند كتابة الأوامر التالية :

```
today = DATE()
IF today = " 01/11/88 "
```

يتوقف البرنامج وذلك لأن المتغير (today) يحتوى على قيمة تاريخية فى حين القيمة الموجودة يمين علامة التساوى ("01/11/88") هى قيمة حرفية. فإذا أريد علاج هذا الخطأ يتم تحويل نوع أحد القيمتين إلى نوع القيمة الأخرى كما سيتم الإيضاح فى الجزء الخاص بتحويل القيم التاريخية.

٢٩ - ١ - ٤ إستخدام الدالة (LEN)

فى بعض الأحيان يحتاج مخطط البرامج إلى تحديد طول سلسلة حرفية(String). فمثلا عندما يقوم المستخدم بإدخال بيانات معينة يمكن اختبار طول السلسلة الحرفية (String) التى قام بكتابتها للتأكد من صحة هذه البيانات وتستخدم لذلك الدالة (LEN()) وهذه الدالة ينتج عنها قيمة عددية تمثل طول هذه السلسلة. فمثلا إذا كان هناك متغير عددى (String) عدد حروفه ٢٠ حرفا فعند استخدام الأمر التالى :

```
? LEN(string)
```

والضغط على مفتاح الإدخال يظهر على الشاشة الرقم ٢٠.

وعند كتابة الأمر التالى :

```
? LEN("Mohamed").
```

والضغط على مفتاح الإدخال يظهر الرقم (7).

والأوامر التالية توضح استخدام هذا الأمر داخل برنامج.

```
IF LEN (Input) < 5
   Do Error & & Branch to Error.prg
ELSE
   ------
   ------
   ------    commands
   ------
   ------
ENDIF
```

وفى هذه الأوامر يقوم البرنامج باختبار طول السلسلة الحرفية الموجودة فى المتغير (Input) فإذا كان أقل من (5) يتم التفرع إلى البرنامج (Error).

٢٩ - ١ - ٥ إستخدام الدالة (SUBSTR)

تستخدم هذه الدالة للحصول على جزء من سلسلة الحروف (String). ويجب فى هذه الحالة إبلاغ البرنامج عن مكان البداية وطول السلسلة المطلوب الحصول عليها. فمثلا إذا كان هناك متغير إسمه (String1) يحتوى على الحروف التالية :

" My name is HASAN "

فعند كتابة الأمر التالى والضغط على مفتاح الإدخال :

```
? SUBSTR(string1,1,10)
```

يلاحظ ظهور الآتى :

```
My name is
```

وذلك لأن العدد الأول (1) يحدد بداية السلسلة المطلوبة والعدد الثانى (10) يحدد عدد حروف هذه السلسلة. ويجب ملاحظة أن المسافات الخالية أيضا تؤخذ فى الإعتبار.

وعند كتابة عدد واحد بعد الدالة فإن ذلك يعنى أن السلسلة المطلوبة تبدأ من هذا العدد وتنتهى بنهاية السلسلة الأصلية. فمثلا عند كتابة الأمر التالى والضغط على مفتاح الإدخال :

```
? SUBSTR ( string1,12 )
```

يلاحظ ظهور الآتى :

```
HASAN
```

ويمكن استخدام هذه الدالة فى إعادة استخدام سلسلة من الحروف فى عدة أماكن من البرنامج. وذلك عن طريق تخزين هذه السلسلة فى متغير ذاكرة واستخدام أى جزء منها حسب الحاجة. فمثلا فى المثال السابق عندما يراد استخدام جزء من السلسلة مع إسم آخر يستخدم الأمر التالى :

```
? SUBSTR(string1,1,10) +"MOHAMED"
```

فعند الضغط على مفتاح الادخال ، يلاحظ ظهور الآتى :

```
My name is MOHAMED
```

ويلاحظ فى هذه الحالة ظهور الإسم(MOHAMED) مكان الإسم(HASAN).

٢٩ - ١ - ٦ الدالــة (LEFT) والدالــة (RIGHT)

تستخدم الدالة (LEFT()) للحصول على جزء من سلسلة حرفية بدءا من يسار السلسلة وبعدد معين من الحروف. فمثلا فى المثال السابق يمكن استخدام الدالة كالآتى :

```
? LEFT(string1,10)
```

وعند الضغط على مفتاح الإدخال يظهر الآتى :

My name is

أما الدالة (RIGHT()) فتؤدى إلى الحصول على جزء من سلسلة حرفية بدءا من يمين السلسلة وبعدد الحروف المحدد بالرقم الموجود مع الدالة. فمثلا فى المثال السابق يمكن استخدام الدالة كالآتى :

```
? RIGHT ( string 1 , 5 )
```

وعند الضغط على مفتاح الإدخال يلاحظ ظهور الآتى :

HASAN

٢٩ - ١ - ٧ إستخدام الدالة (AT)

تستخدم هذه الدالة فى تحديد مكان مجموعة من الحروف داخل سلسلة حرفية. فمثلا فى المثال السابق عندما يراد تحديد مكان كلمة (HASAN) داخل سلسلة حرفية يمكن كتابة السطر التالى :

```
? AT ( 'HASAN' , string1 )
```

وعند الضغط على مفتاح الإدخال يظهر الرقم (12) وهو يعنى أن الإسم يبدأ من الحرف رقم ١٢.

ويلاحظ أن الحروف المطلوب تحديد مكانها توضع بين علامات تنصيص (Quotation).

٢٩ - ١ - ٨ إستخدام الدالة (UPPER) والدالة (LOWER)

تستخدم الدالة (UPPER()) فى تحويل سلسلة حرفية من حروف صغيرة (Lowercase) إلى حروف كبيرة(Uppercase). كما تستخدم الدالة (LOWER())

فى عمل العكس أى تحويل الحروف الكبيرة إلى حروف صغيرة. فمثلا عند استخدام الدالة (UPPER()) مع المتغير الحرفى (string1) كالآتى :

```
? UPPER ( string1 )
```

يلاحظ ظهور الآتى :

```
MY NAME IS HASSAN
```

كما يمكن جعل أول حرف كبيرا (Uppercase) وباقى الحروف صغيرة. وذلك بكتابة السطر التالى :

```
' M '  +  LOWER ( SUBSTR ( string1 , 2 )
```

وعند الضغط على مفتاح الإدخال يلاحظ ظهور الآتى :

```
My name is hasan
```

ويلاحظ من هذا المثال أنه تم إضافة الحرف (M) إلى جزء من السلسلة الحرفية الموجودة فى المتغير (string1) يبدأ من الحرف رقم (2) وهو (y) وحتى آخر السلسلة الحرفية. كما تم تحويل هذا الجزء إلى حروف صغيرة (Lowercase) باستخدام الدالة (LOWER).

والمثال التالى يوضح استخدام الدالة (UPPER) داخل برنامج :

```
@ 15,15 SAY "Send report to printer Y/N" ;
   GET Answer
IF UPPER(Answer) = "Y"
   SET PRINT ON
ENDIF
```

وفى هذا المثال يتم سؤال المستخدم إذا كان يريد طباعة التقرير أم لا. فإذا كان يريد الطباعة فإنه يكتب (y). وفى هذه الحالة إذا كتب المستخدم حرف (y) صغيرا

أو كبيرا لايؤثر ذلك على البرنامج. وذلك لأن الدالة (UPPER) تقوم بتحويل هذا الحرف إلى حرف كبير (Uppercase).

٢٩ - ١ - ٩ إستخدام الدوال (TRIM) ، (LTRIM) ، (RTRIM)

عند إدخال مدخلات حرفية فى بعض الحقول أو متغيرات الذاكرة ففى معظم الأحيان يكون طول السلسلة الحرفية التى يتم إدخالها أصغر من طول الحقل أو متغير الذاكرة. وفى هذه الحالة تكون هناك مسافات خالية (Spaces) بعد السلسلة الحرفية. فمثلا قد يتم إدخال إسم مثل ("Tarek Mohmoud") فى حقل أو متغير ذاكرة طوله ٢٠ حرفا. فى هذه الحالة تبقى مسافات خالية بعد الإسم. وفى معظم الأحيان يريد مخطط البرامج التخلص من هذه المسافات الخالية والحصول على الحروف فقط. وذلك عندما يريد كتابة تقرير مثلا وضم مجموعة من الحقول فى سطر واحد أو عندما يتم إنشاء حقل للإسم الأول وحقل للإسم الثانى ففى هذه الحالة يجب التخلص من المسافات الخالية من حقل الإسم الأول حتى يظهر الإسم بصورة مقبولة. وفى هذه الأحوال وغيرها يتم استخدام الدالة (() TRIM). هذه الدالة تؤدى إلى التخلص من المسافات الخالية بعد نهاية السلسلة الحرفية. ولتوضيح تأثير هذه الدالة يتم دراسة المثال الآتى :

نفرض أنه تم إنشاء حقل للإسم الأول إسمه (Fname) وحقل للإسم الثانى إسمه (Sname). ونفرض أن الحقل (Fname) طوله عشرة حروف والحقل (Sname) طوله عشرة حروف. فعند عرض إسم مثل (Aly Hasan) مثلا يتم كتابة السطر التالى :

```
? Fname + Sname
```

وعند الضغط على مفتاح الإدخال يلاحظ ظهور الإسم كالآتى :

```
Aly        Hasan
```

ويحدث هذا لأن الحقل (Fname) طوله عشرة حروف وتم كتابة ٣ حروف فقط منه أى أن هناك سبعة مسافات خالية (Sapces). فى حين يمكن استخدام الدالة (TRIM) كالآتى :

```
? TRIM ( Fname ) + Sname
```

وعند الضغط على مفتاح الإدخال يلاحظ ظهور الآتى :

AlyHasan

ويلاحظ عدم وجود أى مسافة بين الإسمين. فإذا أريد كتابة الإسمين وبينهما مسافة يتم كتابة السطر التالى :

? TRIM (Fname) + " " + Sname

وعند الضغط على مفتاح الإدخال يلاحظ ظهور الآتى :

Aly Hasan

أما الدالة (LTRIM()) فتستخدم فى التخلص من المسافات الخالية فى أول السلسلة الحرفية. ويفيد هذا فى عدة حالات منها على سبيل المثال التحقق من صحة إدخال المستخدم للمدخلات الحرفية وعدم إدخال مسافة خالية فى بداية أى سلسلة حرفية. حيث أن كتابة مسافة خالية فى أول أى حقل حرفى يمكن أن تسبب مشاكل كثيرة فى الإسترجاع أو البحث عن سجل معين.

فمثلا فى المثال السابق نفرض أن المستخدم عند كتابته الإسم (ALY) قام بالضغط على مسطرة المسافات قبل الإسم عن طريق الخطأ. فى هذه الحالة يظهر الإسم كالآتى مثلا :

-ALY - - - - - - -

فى هذه الحالة يتم تخزين الإسم وفى أوله مسافة خالية (Space). وعندما يراد التخلص من هذه المسافة يستخدم الأمر (LTRIM) كالآتى :

? LTRIM (Fname)

وعند الضغط على مفتاح الإدخال يظهر الإسم كالآتى :

ALY - - - -

أما الدالة (RTRIM) فإنها تؤدى نفس عمل الدالة (TRIM) لأنها تؤدى إلى التخلص من المسافات الموجودة يمين سلسلة الحروف.

ملاحظة

عندما يراد التأكد من التخزين الصحيح للبيانات التى يكتبها المستخدم ، والتغلب على الأخطاء التى قد تنتج عن كتابة مسافات خالية فى أول أو آخر الإسم. يمكن استخدام متغير الذاكرة أولا فى استقبال مدخلات المستخدم. ثم يتم استخدام الدالة (TRIM) والدالة (LTRIM) مع هذا المتغير. وذلك قبل نقله فعلا الى مكانه فى الملف.

فمثلا إذا كان هناك حقل للإسم (name) يتم إنشاء متغير ذاكرة إسمه (mname) وذلك كالآتى :

```
STORE SPACE(30) TO mname
```

ثم يتم كتابة السطر التالى للتخلص من المسافات فى أول الإسم وآخره كالآتى :

```
STORE LTRIM ( TRIM (mname) )  TO mname
```

فى هذه الحالة يتم تخزين ما يدخله المستخدم فى متغير الذاكرة (mname) بدون أى مسافات فى أوله أو آخره.

وتستخدم الدالة (TRIM) أيضا فى الحصول على الطول الصحيح لأى سلسلة حرفية باستخدام الدالة (LEN). حيث أن مجرد استخدام الدالة (LEN) يمكن أن يعطى نتيجة خاطئة نتيجة وجود مسافات خالية (Spaces) فى أول السلسلة الحرفية أو فى آخرها ولكن مثلا عند كتابة الآتى :

```
LEN  ( LTRIM ( TRIM ( mname ) ) )
```

فى هذه الحالة يتم الحصول على الطول الحقيقى لهذا الإسم بدون أى مسافات فى أوله أو فى آخره.

٢٩ - ١ - ١٠ جمع البيانات الحرفية (CONCATINATION)

يمكن جمع بيانات حرفية (String) على بيانات حرفية أخرى باستخدام علامة الجمع (+). ويؤدى هذا إلى تكوين سلسلة حرفية جديدة (String) تحتوى على السلسلة الحرفية الأولى يليها السلسلة الحرفية الثانية. فمثلا عندما يكون هناك حقل حرفى إسمه (Account_No) ويراد جمع بيانات الحقل على سلسلة حرفية أخرى (String) يمكن استخدام الأمر التالى :

```
" The account number is " + Account_no ?
```

فإذا افترضنا أن الحقل (Account_No) يحتوى على العدد الآتى (5788) فإن تنفيذ الأمر السابق يؤدى إلى ظهور الآتى :

```
The account number is 5788
```

ويلاحظ فى هذه الحالة التصاق السلسلة الحرفية الأولى بالسلسلة الحرفية الثانية بدون أى مسافات.

ويمكن التغلب على ذلك بترك مسافة خالية فى آخر السلسلة الحرفية بعد كلمة (is) كالآتى مثلا :

```
? " The account no is  " + Account_no
```

ويلاحظ فى هذه الحالة إضافة مسافة خالية بعد السلسلة الأولى. كما يمكن تنفيذ ذلك بطريقة أخرى بكتابة الأمر التالى :

```
? " The account no is " + " " + Account_no
```

ويلاحظ فى هذه الحالة جمع سلسلة حرفية أخرى تحتوى على مسافة خالية بين السلسلة الأولى والسلسلة الثانية.

وهناك طريقة أخرى لجمع السلاسل الحرفية (strings) وذلك باستخدام علامة (-). والجمع بواسطة علامة (-) يؤدى إلى نقل المسافات الخالية الخلفية

(Trailing blanks) من السلسلة الأولى إلى نهاية السلسلة الثانية. فمثلا فى المثال الخاص بالإسم عند جمع الحقل الخاص بالإسم الأول (Fname) إلى الحقل الخاص بالإسم الثانى (Sname) باستخدام علامة (-) يلاحظ ضم الإسمين وإنتقال المسافات الخالية من الإسم الأول إلى نهاية الإسم الثانى.

٢٩ - ١ - ١١ التحويل بين الحروف وكود الآسكى

يتيح برنامج (+ DBase III) لمخطط البرامج الحصول على أى حرف عن طريق كود الآسكى (ASCII Code) الخاص بهذا الحرف وذلك باستخدام الدالة (CHR). ويساعد هذا فى حالات كثيرة منها مثلا التحكم فى شكل شاشة الإدخال. حيث يمكن الحصول على بعض الحروف الخاصة التى تساعد على رسم الخطوط والأشكال التى يمكن من خلالها رسم شاشة الإدخال. فمثلا الدالة ((CHR(205) تساعد فى رسم مستطيل على الشاشة وكذلك الدالة ((CHR(201) تساعد فى رسم أركان هذا المستطيل وهكذا.

كما يمكن الحصول على أى حرف عن طريق كود الآسكى الخاص به فمثلا للحصول على حرف (a) يتم كتابة السطر التالى :

```
? CHR ( 97 )
```

وعند الضغط على مفتاح الإدخال يظهر الحرف (a).

كذلك يمكن الحصول على كود الآسكى (ASCII Code) الخاص بأى حرف باستخدام الدالة (ASC). فمثلا للحصول على كود الآسكى الخاص بالحرف (a) يتم كتابة السطر التالى :

```
? ASC ( 'a' )
```

وعند الضغط على مفتاح الإدخال يظهر العدد (97). كما يمكن الحصول على أى حرف عن طريق حرف آخر كالآتى مثلا :

```
? CHR ( ASC ('a') + 1 )
```

وعند الضغط على مفتاح الإدخال يظهر الحرف (b) وذلك لأن كود الآسكى الخاص بالحرف (b) يزيد عن كود الآسكى الخاص بالحرف (a) بمقدار (1). وكذلك بالنسبة لجميع الحروف الهجائية حيث أن كل حرف يزيد كود الآسكى الخاص به عن الحرف الذى يسبقه بمقدار (1).

كما أن هناك بعض أرقام الآسكى (ASCII Code) التى تؤدى إلى حدوث تأثيرات معينة مثل تشغيل الجرس (Bell). فمثلا عند كتابة الأمر التالى والضغط على مفتاح الإدخال

```
? CHR(7)
```

يلاحظ تشغيل الجرس (Bell).

ويمكن عن طريق ذلك توجيه إنتباه المستخدم عند حدوث خطأ مثلا أو فى أى حالات أخرى مشابهة.

٢٩ - ٢ الدوال العددية

هناك عدة دوال عددية يتيح برنامج (+ DBase III) لمخطط البرامج استخدامها داخل البرنامج ويتم إلقاء الضوء عليها فى الأجزاء التالية.

٢٩ - ٢ - ١ الدالة (ABS)

وتستخدم هذه الدالة فى الحصول على القيمة المطلقة لأى عدد. ويمكن استخدامها فى تحديد الفرق العددى بين قيمتين عدديتين دون الحاجة لمعرفة أيهما أكبر من الأخرى. فمثلا يمكن ملاحظة الأوامر التالية :

```
i = 20
j = 80
? ABS(i-j)
```

وعند الضغط على مفتاح الإدخال يلاحظ ظهور العدد ٦٠.

٢٩ - ٢ - ٢ الدالـــة (EXP)

وتستخدم هذه الدالة فى الحصول على القيمة الأسية (e). فمثلا للحصول على النسبة التقريبية (e) يتم كتابة الأمر التالى :

? EXP(1)

والضغط على مفتاح الإدخال فيلاحظ ظهور العدد الآتى :

(2.718)

٢٩ - ٢ - ٣ الدالـــة (INT)

وتستخدم هذه الدالة فى الحصول على العدد الصحيح فى قيمة عددية معينة. فمثلا عند تخزين العدد (15.52) فى المتغير (x) ثم كتابة الأمر التالى والضغط على مفتاح الإدخال :

? INT (X)

يلاحظ ظهور العدد (15)

٢٩ - ٢ - ٤ الدالـــة (LOG)

وتستخدم هذه الدالة فى الحصول على اللوغاريتم الطبيعى لأى عدد. وهى نادرا ما تستخدم فى برامج قواعد البيانات ولكن قد تكون هناك بعض التطبيقات الرياضية التى تتطلب استخدامها.

٢٩ - ٢ - ٥ الدالـــة (MAX)

وتستخدم هذه الدالة فى الحصول على أكبر قيمة من قيمتين ويمكن استخدامها فى تحديد القيم التى تزيد عن قيمة معينة. وذلك عن طريق تحديد قيمة معينة يراد اعتبارها الحد الأدنى للقيم الموجودة فى حقل معين. ثم مقارنة جميع القيم الموجودة فى الحقل بالقيمة التى تم تحديدها للحصول على جميع القيم التى تزيد عن هذه القيمة.

٢٩ - ٢ - ٦ الدالـــة (MIN)

وتستخدم هذه الدالة فى الحصول على أصغر قيمة من قيمتين. ويمكن استخدامها فى تحديد الحد الأعلى للقيم الموجودة فى حقل معين وتحديد القيم التى تقل عن هذا الحد.

٢٩ - ٢ - ٧ الدالـــة (MOD)

وتستخدم هذه الدالة لتحديد المقدار الباقى بعد قسمة عدد على عدد آخر كما تستخدم فى التحويل من نوع من الوحدات إلى نوع آخر. فمثلا لتحويل عدد من الدقائق يساوى ٣٦٥٠٠ إلى ما يقابله من أيام وساعات ودقائق يمكن استخدام الأوامر التالية :

```
t = 36500
minutes = MOD(t,60)
h = INT(t/60)
hours = MOD(h,24)
days = INT(h/24)
? t,"minutes are",days,"days",hours,"hours";
minutes,"minutes"
```

وعند الضغط على مفتاح الإدخال يظهر الآتى :

```
36500 minutes are : 25 days 8 hours 20 minutes
```

وذلك لأن السطر الثانى (MOD (t , 60)) يؤدى إلى ظهور باقى قسمة العدد (36500) على (60).

والسطر الثالث (INT (t/60)) يؤدى إلى ظهور العدد الصحيح الناتج عن قسمة العدد (36500) على 60 أى عدد الساعات الكلية.

والسطر الرابع (MOD (h,24)) يؤدى إلى حساب العدد الباقى من قسمة عدد الساعات الكلية على (24) أى عدد الساعات المتبقية من الأيام.

والسطر الخامس (INT (h/24)) يؤدى إلى ظهور عدد الأيام الصحيحة.

والسطر السادس يؤدى إلى عرض الأعداد الممثلة للأيام والساعات والدقائق.

٢٩ - ٢ - ٨ الدالة (ROUND)

وتستخدم هذه الدالة لعمل تقريب للكسر العشرى بعد تحديد عدد معين من الكسور العشرية المطلوب ظهورها فى العدد. فمثلا عند كتابة الأمر التالى :

```
?   ROUND(15.847321,2)
```

والضغط على مفتاح الادخال يظهر الآتى :

```
15.85
```

٢٩ - ٢ - ٩ الدالة (SQRT)

وتستخدم هذه الدالة للحصول على الجذر التربيعى لأى عدد.

٢٩ - ٣ الدوال التاريخية (Date Functions)

يتعامل البرنامج مع التواريخ عن طريق تمثيل كل تاريخ بعدد معين. ورغم أن البرنامج يعرض التاريخ على الشاشة على الشكل المعروف بالنظام الأمريكى (MM/DD/YY) والذى يعنى رقمين للشهر (MM) ورقمين لليوم (DD) ورقمين للسنة (YY) إلا أن البرنامج يتعامل مع العدد الممثل لهذا التاريخ فقط. وعن طريق ذلك العدد يمكن طرح تاريخ من تاريخ للحصول على عدد الأيام المحصورة بين التاريخين. كما يمكن إضافة عدد من الأيام إلى تايخ معين للحصول على تاريخ جديد أو طرح عدد من الأيام من تاريخ معين للحصول على تاريخ جديد وهكذا.

كما يوفر البرنامج عددا من الدوال التى تستخدم فى التعامل مع التواريخ مثل الدالة (DATE()). هذه الدالة تعطى دائما تاريخ اليوم الحالى الذى يتم إدخاله عند بدء تشغيل نظام التشغيل (MS-DOS).

فمثلا عندما يكون هناك ملف حسابات (Accounts) وهناك مجموعة من العملاء الذين يحل ميعاد دفع الدين الخاص بهم فى يوم محدد وليكن هذا اليوم هو (01/01/1990) وهو اليوم الحالى. فيمكن كتابة السطر التالى :

```
LIST FOR Date = DATE()
```

وفى هذه الحالة تظهر قائمة بأسماء الأشخاص الذين يحل موعد سدادهم الدين فى هذا اليوم. كما يمكن تحديد موعد سداد قرض بعد خمسين يوما من اليوم الحالى كالآتى مثلا :

```
STORE DATE() + 50 TO overdue
```

حيث يتم إنشاء متغير ذاكرة (Overdue) يتم تخزين موعد سداد القرض فيه ثم اختبار تاريخ السداد بعد خمسين يوما عن طريق السطور التالية من البرنامج :

```
IF overdue = DATE()
Do letter
ENDIF
```

وفى هذه الحالة يتم تشغيل برنامج (Letter) عندما يكون تاريخ اليوم الحالى (Date()) مساويا لتاريخ الدفع (Overdue).

وهناك دوال تساعد على تحديد ترتيب اليوم فى الأسبوع أو فى الشهر أو تحديد ترتيب الشهر فى السنة أو تحديد السنة نفسها. فمثلا بالنسبة للتاريخ (01/28/1990) يمكن الحصول على ترتيب اليوم فى الأسبوع كالآتى :

```
? DOW ( DATE () )
```

حيث تعطى الدالة (DOW) اليوم المقابل للتاريخ. فعند الضغط على مفتاح الإدخال يلاحظ ظهور العدد (1) الذى يدل على أن اليوم هو الأحد حيث أن يوم الأحد يمثل أول أيام الأسبوع. كما يمكن الحصول على ترتيب اليوم فى الشهر كالآتى :

```
? DAY ( DATE() )
```

وعند الضغط على مفتاح الإدخال يظهر العدد (28) الذى يدل على أن اليوم هو الثامن والعشرون من الشهر.

كما يمكن الحصول على ترتيب الشهر فى السنة كالآتى :

```
? MONTH ( DATE() )
```

وعند الضغط على مفتاح الإدخال يظهر العدد (1) الذى يدل على أن الشهر هو شهر يناير.

كما يمكن الحصول على السنة كالآتى :

```
? YEAR ( DATE() )
```

وعند الضغط على مفتاح الادخال يظهر العدد (1990)

٢٩ - ٣ - ١ تحويل التاريخ إلى حروف (DATE TO CHARACTER)

كما سبق الإيضاح فإن التواريخ يتم تمثيلها فى البرنامج كأعداد. فعندما يراد استخدام هذه التواريخ فى سلاسل حرفية (Strings) فى هذه الحالة يلزم أولا تحويل هذه التواريخ إلى حروف (Characters). وتستخدم لذلك الدالة (DTOC()).

فمثلا عند كتابة السطر التالى :

```
? DTOC(DATE())
```

والضغط على مفتاح الادخال يلاحظ ظهور التاريخ الحالى كالآتى :

```
01/28/90
```

تحـذير

لايمكن إجراء حسابات على التاريخ وهو فى صورة سلسلة حرفية (String) مثل إضافة أو طرح عدد من الأيام من التاريخ أو طرح تاريخ معين من هذا التاريخ.

ولكن يلزم أولا تحويله مرة ثانية إلى تاريخ غير حرفى كما سيتم الإيضاح فيما بعد.

وهناك دوال أخرى يمكن استخدامها فى عرض أيام الأسبوع بالحروف بدلا من الأرقام كالآتى مثلا :

```
? CDOW (DATE())
```

وعند الضغط على مفتاح الإدخال يظهر الآتى :

```
Sunday
```

كما يمكن عرض الشهور بالحروف أيضا كالآتى :

```
? CMONTH(DATE())
```

وعند الضغط على مفتاح الإدخال يظهر الآتى :

```
January
```

وتستخدم هذه الدوال فى عرض التواريخ بأى صورة مطلوبة من خلال البرنامج. فمثلا يمكن كتابة السطر التالى :

```
? CDOW(DATE()) +',' + CMONTH(DATE()) +' '+;
LTRIM(STR(DAY(DATE()),2))+','+STR(YEAR(DATE()),4)
```

وعند الضغط على مفتاح الإدخال يظهر الآتى :

```
Sunday , January 12,1990
```

وعند إدخال هذا الأمر فى البرنامج فإنه ينفذه على تاريخ اليوم الحالى حسب تاريخ اليوم الذى يتم فيه تشغيل البرنامج.

٢٩ - ٣ - ٢ تحويل الحروف إلى تاريخ (CHARACTER TO DATE)

عندما يراد استخدام التواريخ الحرفية كتواريخ مفهومة بالنسبة لبرنامج (+ DBase III) وذلك حتى يمكن استخدام هذه التواريخ فى العمليات الحسابية المختلفة كما سبق الإيضاح فى هذه الحالة تستخدم الدالة (CTOD()) التى تقوم بتحويل التاريخ الحرفى إلى التاريخ المقابل. فمثلا عند كتابة الأوامر التالية :

```
STORE '1/20/90' TO string
STORE CTOD ( string ) TO  newday
```

فالسطر الأول يؤدى إلى تكوين متغير ذاكرة إسمه (string) يحتوى على السلسلة الحرفية (1/20/90).

أما السطر الثانى فإنه يؤدى إلى تكوين متغير تاريخى نتيجة تحويل المتغير الحرفى إلى متغير تاريخى.

ويجب ملاحظة أن المتغير (newday) فى هذه الحالة يحتوى على نفس الأرقام (1/20/90) ولكنها تمثل شيئا مختلفا عن الأرقام الموجودة فى متغير الذاكرة الحرفى (string).

وعندما يراد إنشاء متغير تاريخى غير تاريخ اليوم يمكن استخدامه داخل البرنامج يستخدم السطر التالى :

```
Date = CTOD('01/10/90')
```

ويمكن بعد ذلك إدخال أى تاريخ فى هذا المتغير كما يمكن إنشاء متغير تاريخى خال (blank) كالآتى :

```
Date1 = CTOD ( ' // ' )
```

٢٩ - ٣ - ٣ استخدام التواريخ فى المقارنة (Comparison)

عند عمل مقارنة بين تاريخين فإن هذين التاريخين يجب ألا يكونا حرفيين. أى يلزم أولا تحويلهما إلى تاريخ ثم عمل المقارنة المطلوبة. فمثلا عند كتابة السطر التالى :

? ' 01/01/90 ' > ' 12/31/89 '

وعند الضغط على مفتاح الإدخال فإن ذلك يعطى (.F.) أى غير صحيح مع أن التاريخ (12/31/89) يسبق التاريخ (01/01/90) أى أن النتيجة يجب أن تكون صحيحة (True). لذلك يجب أولا تحويل التاريخ الحرفى قبل تنفيذ عملية المقارنة كالآتى :

? CTOD('O1/01/90') > CTOD('12/31/89')

وعند الضغط على مفتاح الإدخال تظهر النتيجة (.T.) أى صحيح (True).

وعندما يراد مثلا عرض بيانات الأشخاص الذين يحل موعد سدادهم الدين فى التاريخ (01/28/90) يتم كتابة السطر التالى :

DISPLAY ALL FOR overdue = CTOD ('01/28/90')

٢٩ - ٣ - ٤ إستخدام الدالة (TIME())

يستخدم البرنامج الدالة (TIME()) لإعطاء الوقت الحالى على هيئة (hours : minutes : seconds) وهو الوقت الذى يتم إدخاله عند بدء تشغيل الجهاز. فمثلا عند كتابة الأمر التالى :

? TIME()

والضغط على مفتاح الإدخال يظهر الوقت الآتى مثلا :

10 : 30 : 35

ولتخزين الوقت الحالى داخل حقل قاعدة البيانات يتم تعريف هذا الحقل بطول (٨) حروف ثم يتم استبدال مكونات هذا الحقل بالوقت الحالى. فمثلا إذا كان هناك حقل فى ملف قاعدة البيانات إسمه (Now) يمكن كتابة السطر التالى :

REPLACE Now WITH TIME()

فى هذه الحالة يتم إدخال الوقت الحالى فى الحقل (Now) وليكن كالآتى مثلا :

10:30:35

ويمكن عرض جزء من الوقت مثلا يتضمن الساعات والدقائق فقط ولتنفيذ ذلك يتم كتابة السطر التالى :

```
REPLACE Now WITH  SUBSTR( Now , 1 , 5 )
```

وفى هذه الحالة يحتوى المتغير (Now) على الوقت الآتى :

10 : 30

ملاحظة

ماسبق ذكره فى هذا الفصل ينطبق أيضا على كل برامج عائلة (DBase) مثل (DBase IV) ، (FoxBase) ، (FoxBase +) ، (FoxPro).

الفصل الثلاثون

مزيد من التحكم فى شاشة الإدخال

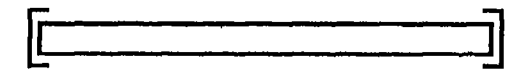


كما سبق الإيضاح فى الفصل الخاص بالتحكم فى شاشة الإدخال فإن برنامج (DBase III +) يتيح لمخطط البرامج التحكم فى الرسائل التى تظهر على الشاشة وشكل الحقول التى تظهر على الشاشة باستخدام الأمر (@...GET) والأمر (@...SAY) ... وهكذا. وفى هذا الفصل يتم عرض وسائل أخرى للتحكم فى الشاشة تتيح لمخطط البرامج تصميم شاشات أحسن وأكثر وضوحا.

٣٠ - ١ التحكم فى شكل العمود الضوئى (Highlight)

الوضع المبدئى للأوامر (APPEND, EDIT, GET) هو ظهور أعمدة ضوئية تمثل الحقول المطلوبة ويحدد كل عمود ضوئى طول الحقل الذى يمثله. واذا أراد المستخدم استخدام علامات تحديد (Delimiters) للبيانات الموجودة فى الحقل فإنه يستخدم الأمر (SET DELIMITERS ON) حيث أن الوضع المبدئى لهذه العلامات (DELIMITERS) يكون (OFF).

وعند استخدام الأمر (SET DELIMITERS ON) تظهر علامات (Colons) حول بيانات الحقول. ويمكن استخدام أى علامات أخرى بدلا من علامات (Colons) وذلك باستخدام الأمر (SET DELIMITERS TO) ثم كتابة العلامات المطلوب إظهارها بين علامات تنصيص (Quotation). ويجب ملاحظة أن استخدام علامات التحديد (Delimiters) لاتمنع من ظهور العمود الضوئى وذلك كالآتى مثلا :



وإذا أراد مخطط البرامج إلغاء العمود الضوئى فإنه يستخدم الأمر (SET INTENSITY OFF).

٣٠ - ٢ استخدام العناوين النسبية

كما سبق الإيضاح فإن استخدام الأمر (@...SAY) والأمر (@...GET) يتيح لمخطط البرامج عرض البيانات أو الرسائل فى أى مكان على الشاشة حسب الإحداثيات التى يتم كتابتها بعد الحرف (@). ولكن فى بعض الأحيان لايكون المكان المطلوب الكتابة فيه محددا. فمثلا عندما يراد عرض بيانات بعض السجلات ثم عرض رسالة

للمستخدم بعد آخر سطر فى البيانات بسطرين مثلا فى هذه الحالة فإن مخطط البرامج لايعرف مقدما عدد السجلات التى سيتم عرض بياناتها. وبالتالى لايعرف المكان الذى يجب عرض الرسالة فيه. لذلك تستخدم الدالة (ROW()) والدالة (COL()) لتحديد العناوين النسبية. وهذا يعنى أن إحداثيات نقطة معينة على الشاشة تعتمد على إحداثيات آخر نقطة تم الوصول إليها.

والدالة (ROW()) مثلا تعطى رقم السطر الذى يقف عنده المؤشر فى هذه اللحظة. فمثلا عند كتابة السطر التالى :

```
@ 5,3  SAY  ' Hello Mohamed '
```

فى هذه الحالة فإن السطر الحالى (Current row) هو السطر رقم (5) كما أن العمود الحالى (Current column) هو العمود رقم ٣. فعند كتابة الأمر التالى :

```
@ ROW() + 2,3  SAY ' How are you? '
```

ثم تنفيذ هذين الأمرين يظهر الآتى على الشاشة :

```
Hello Mohamed

How are you?
```

ويلاحظ فى هذه الحالة ظهور الرسالة الثانية بعد سطرين من الرسالة الأولى. وكذلك عند كتابة السطر التالى :

```
@ ROW()+3,COL()+2 SAY 'Well, thank you'
```

وعند تنفيذ الأوامر الثلاثة معا يظهر الآتى على الشاشة :

```
Hello Mohamed

How are you?

                   Well, thank you
```

وفى هذه الحالة يلاحظ ظهور الرسالة الثالثة بعد ثلاثة سطور من الرسالة الثانية وبعد عمودين من آخر حرف فى الرسالة الثانية.

ملاحظة

لايمكن استخدام الدالة (ROW()) والدالة (COL()) بعد الأمر (READ) مباشرة. لأن الأمر (READ) يضع المؤشر على السطر رقم ٢٣ والعمود رقم صفر.

ويمكن استخدام العناوين النسبية أيضا فى عرض علامات معينة بعد مايكتبه المستخدم. فمثلا عندما يراد عرض علامة (*) بعد الرسالة (Message) التى يكتبها المستخدم يستخدم الأمر التالى :

```
@ 6,24 + LEN(message)  SAY  " * "
```

كما يمكن استخدام متغيرات الذاكرة (Memory Variables) فى التحكم فى العناوين النسبية وذلك كالآتى مثلا :

```
Line = 4
DO WHILE .NOT. EOF()
    @ Line,10 SAY Name
    @ Line,40 SAY Address
    Line = Line + 1
    SKIP
ENDDO
```

فى هذه الحالة يتم عرض بيانات السجل الأول على السطر الرابع ثم عرض بيانات السجل الثانى على السطر الخامس وهكذا.

٣٠ - ٣ ضبط الحروف فى المنتصف (Centering A String)

عندما يراد كتابة السلسلة الحرفية إبتداء من منتصف السطر فيمكن ببساطة استخدام الأمر (SAY...@) مع كتابة إحداثيى نقطة المنتصف بعد الحرف (@). ولكن عندما

يراد وضع ما يكتبه المستخدم فى منتصف السطر فإن هذه الطريقة لاتصلح وذلك لأن مخطط البرامج لايعرف مقدما طول السلسلة الحرفية التى يكتبها المستخدم. وفى هذه الحالة يتم أولا التخلص من المسافات الخالية فى السلسلة الحرفية ثم قسمة طولها على (٢) ثم طرح الناتج من (٤٠) حيث أن العمود رقم (٤٠) يمثل منتصف الشاشة تماما. وبذلك يتم تحديد النقطة التى يبدأ منها كتابة السلسلة حتى تصبح فى منتصف السطر تماما.

فمثلا إذا كان هناك متغير ذاكرة (mname) ويراد عرض هذا المتغير فى منتصف السطر يتم كتابة السطور التالية :

```
mname = TRIM ( mname )
mcenter = 40 - LEN ( mname ) / 2
```

وفى هذه الحالة تم تخزين النقطة التى يجب بدء كتابة الإسم عندها حتى يصبح فى منتصف السطر تماما فى متغير إسمه (mcenter). فلكى تتم كتابة هذا الإسم فى منتصف السطر يمكن كتابة الأمر التالى :

```
@ 10 , mcenter  SAY mname
```

فى هذه الحالة يظهر الإسم فى منتصف السطر بصرف النظر عن طول السلسلة الحرفية.

٣٠ - ٤ ضبط الحروف من اليمين (Right Justifying)

عندما يراد كتابة مجموعة من الرسائل (Messages) بحيث تنتهى كلها عند نقطة ثابتة فى اليمين يتم إنشاء متغير ذاكرة (إسمه width مثلا) ويتم تخزين رقم العمود الثابت (60) مثلا به. ويتم استخدام الأوامر التالية :

```
STORE 60 TO width
@ 5, width - LEN ( message1 ) SAY message1
@ 6, width - LEN ( message2 ) SAY message2
```

فعندما يكون المتغير (message1) محتويا على الرسالة التالية :

```
Enter your name
```

والمتغير (message2) يحتوى على الرسالة التالية :

Enter your address

فعند تنفيذ الأوامر السابقة يظهر الآتى على الشاشة :

Enter your name
Enter your address

٣٠ - ٥ حشر حروف داخل السلسلة الحرفية (Stuffing)

عندما يراد حشر مجموعة من الحروف داخل سلسلة حرفية معينة، فإن ذلك يتم باستخدام الدالة (STUFF()). فمثلا عندما تكون هناك سلسلة حرفية (message1) تحتوى على الرسالة التالية :

" Type Q to quit "

ويراد حشر مجموعة من الكلمات داخل هذه الرسالة حتى تصبح :

" Type R to return to main menu or Q to quit "

فى هذه الحالة يتم أولا تخزين السلسلة الحرفية المطلوب إضافتها فى متغير ذاكرة جديد (message2) مثلا وذلك كالآتى :

message2 = " R to return to main menu or "

ويلاحظ كتابة مسافة خالية فى نهاية السلسلة الحرفية ثم يتم كتابة الأمر التالى :

STUFF(message1 , 6 , 0 , message2)

والمعامل الأول لهذه الدالة (message1) هو الرسالة الأصلية. والمعامل الثانى (6) هو رقم الحرف الذى يبدأ عنده إدخال السلسلة الحرفية الثانية (message2). والمعامل

الثالث (0) يحدد عدد الحروف التى يتم استبدالها من السلسلة الأولى بحروف أخرى من السلسلة الثانية. والصفر فى هذه الحالة يعنى أنه لايتم استبدال أى حرف. لأن السلسلة الثانية مطلوب إضافتها دون حذف أى حروف من السلسلة الأولى. والمعامل الرابع هو السلسلة الحرفية الثانية (message2).

ويمكن تنفيذ نفس هذه العملية بكتابة السطر التالى :

```
SUBSTR(message1,1,5) + message2 + SUBSTR(message1,6)
```

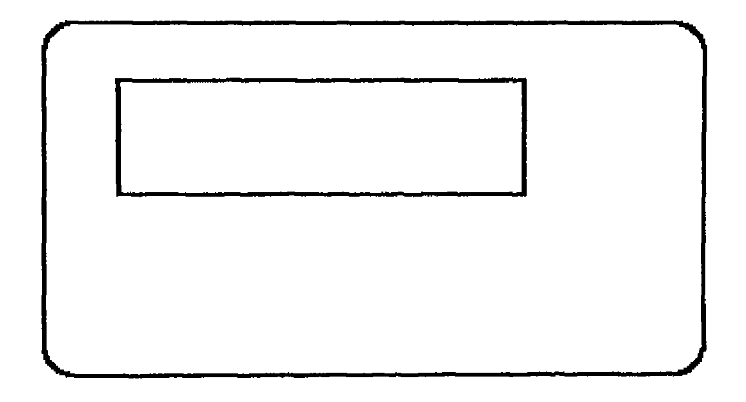
كما يمكن استخدام الأمر (STUFF()) فى استبدال حروف داخل سلسلة حرفية بحروف أخرى. وكذلك فى مسح حروف داخل سلسلة حرفية وذلك باستبدالها بحروف خالية (Spaces).

٣٠ - ٦ رسم الخطوط حول البيانات

يمكن رسم خطوط مفردة (Single) أو خطوط مزدوجة حول البيانات وذلك باستخدام الأمر (@...TO). فمثلا يمكن كتابة السطر التالى :

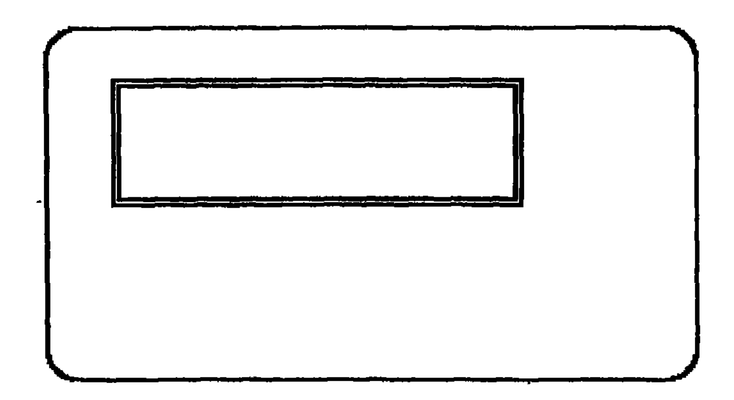
```
@ 2,2 TO 7,50
```

وهذا يؤدى إلى رسم مستطيل يبدأ من النقطة (2,2) إلى النقطة (7,50).



كما يمكن كتابة السطر التالى :

@ 2,2 TO 7,50 DOUBLE



وهذا يؤدى إلى رسم مستطيل بخطوط مزدوجة (Double) يبدأ من النقطة (2,2) إلى النقطة (7,50).

ويمكن رسم خط أفقى مزدوج بكتابة السطر التالى :

@ 2,2 TO 2,79 Double

وذلك بتثبيت رقم السطر.

كما يمكن بالمثل رسم خط رأسى مزدوج بكتابة السطر التالى :

@ 2,2 TO 23,2 DOUBLE

وذلك بتثبيت رقم العمود.

ويمكن مسح المستطيل الذى سبق تكوينه باستخدام الأمر (@...CLEAR TO) وذلك كالآتى :

@ 2,2 CLEAR TO 7,50

٣٠ - ٧ إستخدام ملفات الذاكرة

عندما تكون هناك رسائل يتم كتابتها فى كثير من البرامج فمن المفيد تخزين هذه الرسائل فى متغيرات ذاكرة ثم تخزين هذه المتغيرات فى ملف ذاكرة (Memory File) وذلك لتوفير الوقت والجهد اللازم لكتابتها فى كل برنامج. فمثلا عندما تكون هناك رسالة كالآتى :

```
Send report to printer?(Y/N)
```

فيمكن تخزين هذه الرسالة فى متغير ذاكرة (message1). وعندما يراد عرض هذه الرسالة فى مكان معين على الشاشة يتم كتابة السطر التالى :

```
@ 5,5 SAY message1
```

كما يمكن تخزين أى خطوط أو أشكال ثم عرضها على الشاشة فى أى مكان.

٣٠ - ٨ تكرار الحروف (Repeating Characters)

تستخدم الدالة (REPLICATE()) فى تكرار حرف معين عدة مرات. وتفيد هذه العملية فى رسم أشكال معينة باستخدام هذه الدالة مع الدالة (CHR()) التى تستخدم فى عرض الحروف الخاصة (Special Characters) عن طريق كتابة كود الآسكى الخاص بكل حرف. ويستخدم هذا فى الحصول على أشكال طريفة للشاشة كما سيتم الإيضاح فيما بعد.

ويتم استخدام الدالة (REPLICATE()) كالآتى مثلا :

```
@ 4,4 SAY REPLICATE ( "*" , 75 )
```

ويؤدى هذا إلى ظهور الحرف (*) مكررا ٧٥ مرة إبتداء من النقطة (4 , 4).

٣٠ - ٩ إنشاء ملفات التشكيل (Format Files)

كما سبق الإيضاح فى الجزء الخاص ببرنامج المساعد (Assistant) فإن الأوامر (CHANGE, EDIT, APPEND) تؤدى إلى ظهور شاشة الإدخال حتى يستطيع المستخدم إدخال البيانات أو تعديلها. هذه الشاشة قد تكون هى الشاشة المبدئية (Default) إذا لم يتم إنشاء شاشة إدخال أخرى. أو يتم تصميم شاشة إدخال بالطريقة التى سبق شرحها فى برنامج المساعد (Assistant) ثم فتح الملف الخاص بهذه الشاشة (Format File) باستخدام الأمر (SET FORMAT TO) ثم كتابة إسم الملف. فى هذه الحالة تظهر شاشة الإدخال التى تم تصميمها عند استخدام أى أمر من الأوامر (CHANGE, EDIT, APPEND).

وكما سبق الإيضاح أيضا فإن تصميم الشاشة يتم عن طريق قائمة الإنشاء (CREATE) أو كتابة الأمر (CREATE SCREEN) ثم كتابة إسم الملف من مشيرة النقطة (Dot Prompt). وهذا يؤدى إلى عرض حقول ملف قاعدة البيانات كما يسمح للمستخدم باختيار الحقول المطلوبة وتحديد الأماكن التى يتم وضع الحقول فيها.

وهذه العملية تؤدى إلى إنشاء ملفين أحدهما يسمى ملف الشاشة (Screen File) والذى يتميز بالإمتداد (SCR.) والملف الآخر يسمى ملف التشكيل (Format File) ويتميز بالإمتداد (FMT.).

وعندما يراد عمل أى تعديل فى هذه الشاشة باستخدام الأمر (MODIFY SCREEN) ، فإن هذا التعديل يتم من خلال ملف الشاشة (Screen File). وهذه التعديلات تنتقل آليا إلى ملف التشكيل (Format File).

وهناك طريقة أخرى لإنشاء ملفات التشكيل (Format File) باستخدام ملفات الأوامر (Command Files). ويتم ذلك عن طريق الأمر (MODIFY COMMAND) ثم كتابة إسم الملف المطلوب إنشاؤه مع مراعاة إضافة الإمتداد (FMT.) إلى هذا الإسم وذلك كالآتى مثلا :

```
MODIFY COMMAND F1.FMT
```

وفى هذه الحالة يتم فتح ملف أوامر جديد للكتابة فيه ويتم كتابة الأوامر التى لاتخرج عن أوامر (SAY...@) وأوامر (GET...@) التى تحدد أماكن الحقول على الشاشة.

فمثلا يمكن كتابة السطور التالية :

```
@   1,9  SAY "Name:"
@   1,20 GET Name
@   3,6  SAY "Address:"
@   3,20 GET Address
@   5,1  SAY "Telephone No"
@   5,20 GET Tel_No
```

ويؤدى هذا إلى ظهور الشاشة التالية :

Name : [ ]

Address : [ ]

Telephone No : [ ]

ويلاحظ أن البرنامج لايحتاج إلى الأمر (READ) وذلك لأن الأوامر (CHANGE , EDIT , APPEND) تتولى تخزين البيانات التى يتم إدخالها. ولكن فى حالة الحاجة إلى استخدام عدة شاشات إدخال يستخدم الأمر (READ) فى نهاية كل شاشة.

٣٠ - ١٠ استخدام ملف التشكيل

بعد إنشاء ملف التشكيل يتم فتح هذا الملف للإستخدام عن طريق الأمر (SET FORMAT TO) ثم كتابة إسم الملف السابق إنشاؤه. ويتم ذلك عن طريق كتابة الأوامر التالية مثلا :

```
USE Cadets
SET FORMAT TO F1
APPEND
```

والسطر الأول يؤدى إلى فتح ملف قاعدة البيانات (Cadets).

والسطر الثانى يؤدى إلى فتح ملف التشكيل (F1) السابق إنشاؤه.

والسطر الثالث يؤدى إلى عرض شاشة الإدخال لإدخال البيانات إلى الملف.

ولإغلاق ملف التشكيل يستخدم الأمر(CLOSE FORMAT) أو الأمر (SET FORMAT TO) دون كتابة أى شىء بعده.

ملاحظة

يجب الإهتمام دائما بإغلاق جميع ملفات التشكيل السابق فتحها عند الإنتهاء من استخدامها وذلك لأن عدد الملفات المسموح بفتحها من جميع الأنواع يكون محددا.

٣٠ - ١١ استخدام عدة صفحات للإدخال (Multiple Pages)

يتيح البرنامج لمخطط البرامج إستخدام عدة شاشات إدخال للبيانات وهذا يساعده على تقسيم البيانات على عدة صفحات حتى يستطيع عرض جميع الحقول خصوصا إذا كانت قاعدة البيانات كبيرة وتحتوى على عدد كبير من الحقول. حيث يتم كتابة الأمر (READ) فى كل مكان يراد مسح الشاشة عنده وعرض شاشة جديدة.

وعندما يقوم المستخدم بإدخال البيانات فإن البرنامج ينتقل إلى الشاشة الثانية بمجرد امتلاء الشاشة الأولى. ويمكن للمستخدم فى هذه الحالة الرجوع إلى الشاشة السابقة باستخدام مفتاح (PgUp). كما يمكنه الإنتقال إلى الشاشة التالية باستخدام المفتاح (PgDn).

ملاحظة

مهما زادت شاشات الإدخال فإن عدد أوامر الـ (GET) يجب ألا يزيد عن ١٢٨.

٣٠ - ١٢ التعامل مع حقول الملاحظات (Memo Fields)

يتم تخزين حقول الملاحظات فى ملف آخر منفصل عن ملف قاعدة البيانات. لذلك فإن التعامل معها يختلف عن التعامل مع أى حقل آخر. فمثلا لايمكن تخزين هذه الحقول فى متغيرات الذاكرة (Memory Variables) وأيضا لايمكن عرض هذه الحقول باستخدام الأمر (@..SAY).

وللكتابة فى حقل الملاحظات يتم عرض شاشة التصحيح عن طريق الأمر (EDIT) أو الأمر (CHANGE) فتظهر شاشة الإدخال. وعندما يراد تعديل حقل الملاحظات يتم وضع مؤشر التصحيح على العمود الضوئى الخاص بالملاحظات. ثم بالضغط على مفتاحى (Ctrl-PgDn) يتم فتح حقل الملاحظات لتصحيحه. وعند الإنتهاء يتم الضغط على مفتاحى (Ctrl-PgUp) لتخزين الحقل.

ولتنفيذ هذه العملية من خلال البرنامج يتم أولا إنشاء ملف تشكيل (Format File) لحقل الملاحظات الذى يسمى (Notes) مثلا كالآتى :

```
* Notch.fmt - format file for changing memo field
@ 10,10  SAY "Press <Ctrl> <PgDn>  to edit notes"
@ 12,10  SAY "To save your changes, Press<ctrl> <PgUp> "
@ 14,10  SAY "Press<Return>to return back"
@ 16,10  GET Notes
```

ويلاحظ فى هذا الملف عرض رسائل للمستخدم لتوضح له الخطوات المطلوب اتباعها لفتح حقل الملاحظات والكتابة فيه ثم التخزين. ولتعديل حقل الملاحظات يتم كتابة الأوامر التالية فى البرنامج :

```
USE Cadets
GOTO recnum
SET FORMAT TO Notch
CHANGE NEXT 1 FIELD NOTES
CLOSE FORMAT
USE
```

والسطر الأول فى البرنامج يؤدى إلى فتح ملف قاعدة البيانات الذى يسمى(Cadets).

والسطر الثانى يؤدى إلى الذهاب إلى سجل محدد سبق تخزين رقمه فى متغير الذاكرة (Recnum).

والسطر الثالث يؤدى إلى فتح ملف التشكيل (Notch) الخاص بتعديل حقل الملاحظات.

والسطر الرابع يؤدى إلى ظهور شاشة الإدخال الخاصة بحقل الملاحظات (Notes) والتى يقوم المستخدم من خلالها بإدخال الملاحظات التى يريد إدخالها أو تعديلها إذا كان سبق إدخالها.

والسطر الخامس يؤدى إلى اغلاق ملف التشكيل.

والسطر السادس يؤدى إلى اغلاق جميع الملفات.

ويمكن عرض محتويات حقول الملاحظات على الشاشة باستخدام الأمر (DISPLAY) والأمر (LIST). كما يمكن استخدام الأمر (?) أيضا لنفس الغرض. كما يمكن التحكم فى عرض الملاحظات المعروضة عن طريق كتابة الأمر التالى :

```
SET MEMOWIDTH TO
```

ثم كتابة العرض المطلوب استخدامه. فعندما يراد مثلا عرض الملاحظات بعرض ٦٠ حرفا يتم كتابة الأمر التالى :

```
SET MEMOWIDTH TO 60
```

وفى هذه الحالة يتم عرض الملاحظات فى سطور كل سطر منها طوله ٦٠ حرفا. ويمكن كتابة هذا الأمر فى ملف المواصفات (Config.sys).

٣٠ - ١٣ زيادة مخزن الكتابة المؤقت (TYPEAHEAD BUFFER)

عندما يقوم المستخدم بإدخال البيانات فإن الحروف التى يكتبها تخزن فى مخزن ذاكرة مؤقت (Buffer). وعندما يصل عدد الحروف إلى عدد محدد تنتقل هذه الحروف إلى الملف. والعدد المبدئى (Default) لهذه الحروف هو ٢٠ حرفا كما يمكن زيادة هذا العدد عن طريق الأمر (SET TYPEAHEAD TO) ثم كتابة أى عدد من صفر إلى ٣٢ ألف حسب سعة الذاكرة المتاحة. وكلما كان هذا العدد كبيرا ساعد ذلك على إدخال البيانات أسرع حيث أن ذلك يتيح للمستخدم الكتابة بسرعة أثناء إدخال البيانات.

ملاحظة

الأمر (SET TYPEAHEAD) لايعمل إلا فى حالة (SET ESCAPE ON) لذلك لايفضل استخدامه إلا فى حالات الضرورة حتى يمكن استخدام الأمر (SET ESCAPE OFF).

ملاحظة

ما سبق ذكره فى هذا الفصل ينطبق أيضا على كل برامج عائلة (DBase) مثل (DBase IV) ، (FoxBase) ، (FoxBase +) ، (FoxPro).

الفصل الواحد والثلاثون

إختبار مدخلات المستخدم



عند استقبال أى مدخلات للمستخدم فإنه من الطبيعى والمتوقع أن يخطئ المستخدم ويكتب حروفا قد تكون غير مطلوبة. وفى هذه الحالة قد يؤدى إدخال هذه الحروف إلى توقف البرنامج أو إلى عدم الحصول على المخرجات المطلوبة. ولذلك فمن المفيد التحكم فى مدخلات المستخدم وعدم قبول أى مدخلات غير مسموح بها. ويتم ذلك عن طريق عمل مايشبه الترشيح (Filtering) لهذه المدخلات بحيث لايدخل إلى البرنامج إلا المدخلات الصحيحة أما أى مدخلات أخرى فإنها لاتمر من هذا المرشح ويمكن أن يتم ذلك بعدة طرق يتم إلقاء الضوء عليها فى هذا الفصل.

٣١ - ١ إستخدام الإختيارات العددية (Numeric Choices)

يفضل عند عرض قائمة اختيارات (Menu) للمستخدم أن تستخدم الأرقام فى القائمة ويقوم المستخدم باختيار رقم من هذه القائمة. حيث أن الأرقام يمكن التحكم فيها عن طريق تحديد مدى معين. فمثلا إذا كانت هناك قائمة تحتوى على الإختيارات التالية :

```
1 - Add new records
2 - Edit
3 - Delete
4 - Display
5 - Return
```

فى هذه الحالة يمكن كتابة السطور التالية :

```
Choice = 1
@ 5,5 GET Choice PICTURE '9' RANGE 1 , 5
READ
```

ويتم التحكم فى الرقم الذى يقوم المستخدم بإدخاله عن طريق تحديد مدى لهذا الرقم من (1) إلى (5). وفى حالة إدخال المستخدم لأى رقم لايقع فى هذا المدى فإن البرنامج لايقبله ولاينتقل البرنامج إلى الخطوات التالية إلا بعد ضغط المستخدم على الرقم الصحيح.

٣١ - ٢ توقع احتمالات الخطأ

يمكن بعد أن يقوم المستخدم بإدخال البيانات إعطاؤه الفرصة لاختبار هذه المدخلات والتأكد من صحتها وذلك عن طريق عرض هذه البيانات للمستخدم ثم عرض الرسالة التالية مثلا :

```
Is this correct ? (Y/N)
```

ثم ينفذ البرنامج حلقة تكرارية عن طريق الأمر (DO WHILE) تساعد المستخدم على تصحيح البيانات التى قام بإدخالها وذلك فى حالة كتابة المستخدم (N) بمايفيد عدم صحة البيانات أما فى حالة صحة البيانات فيتم إكمال تنفيذ البرنامج.

ولكن هناك احتمال أن يكتب المستخدم الحرف (Y) كبيرا (Capital) أو صغيرا (Small) وهذا يمكن التغلب عليه عن طريق استخدام الرمز (!) فى الصورة المطلوبة (PICTURE) وذلك عن طريق السطور التالية مثلا :

```
STORE ' ' TO answer
@ 5,0 GET answer PICTURE '!'
READ
```

فى هذه الحالة إذا أدخل المستخدم (y) أو (n) يقوم البرنامج بتحويلها إلى حروف كبيرة (Capital). ولكن ماذا لو أدخل المستخدم أى حرف آخر غير (y) أو (n) ؟ فى هذه الحالة يجب استبعاد أى حرف آخر يدخله المستخدم غير الحرفين (Y) أو (N) ويستخدم لذلك المعامل $ والذى يعنى وجود حرف معين ضمن سلسلة حروف (String).

فمثلا عند كتابة الأمر التالى :

```
DO WHILE .NOT. answer $ 'YN'
```

فإن هذا يؤدى إلى العودة دائما إلى الحلقة التكرارية فى حالة إدخال المستخدم لأى حرف غير (Y) أو (N).

ولتوضيح ذلك يمكن دراسة مجموعة السطور التالية :

```
choice = ' '
    DO WHILE .NOT. choice $ 'YN'
    choice = ' '
    @ 15,15 GET choice PICTURE '!'
    READ
ENDDO
```

والسطر الأول يؤدى إلى إنشاء متغير الذاكرة (Choice).

والسطر الثانى يؤدى إلى دخول الحلقة التكرارية فى حالة إدخال المستخدم لأى حرف آخر غير حرفى (Y) أو (N). وفى حالة إدخال المستخدم للحرف (Y) أو (N) فإن الحلقة التكرارية لايتم تنفيذها ويتم تنفيذ باقى أوامر البرنامج بناء على ذلك. أما إذا أدخل المستخدم (y) صغيرة (Small) أو (n) صغيرة فإن الحلقة التكرارية تنفذ مرة واحدة. لأن هذا الحرف يتم تحويله إلى حرف كبير (Capital) من خلال السطر الرابع عن طريق الصورة (PICTURE). وفى هذه الحالة يتم تنفيذ باقى أوامر البرنامج بناء على اختيار المستخدم إذا كان (y) أو (n).

والسطر الثالث يؤدى إلى مسح محتويات متغير الذاكرة (Choice) فى حالة إدخال المستخدم لأى حرف غير (y) أو (n). فمثلا عند إدخال أى حرف آخر مثل S مثلا فإن البرنامج ينفذ الحلقة التكرارية ويصل إلى السطر الخامس حيث يطلب من المستخدم إدخال حرف. وفى نفس الوقت يجد المستخدم الحرف (S) مكتوبا فى العمود الضوئى (Highlight) وربما يسبب له ذلك شيئا من الإرتباك. لذلك يستخدم السطر الثالث فى مسح محتويات العمود الضوئى (Highlight) المثل للمتغير (Choice) حتى يظهر أمام المستخدم خاليا. وذلك يعنى أن البرنامج لم يتقبل الحرف الذى تم إدخاله.

ملاحظة

يجب ملاحظة الفرق بين المعامل $ والدالة (SUBSTR()) فالمعامل $ يبحث عن حروف معينة فى سلسلة حرفية وإذا وجدها فإنه يعطى القيمة(.T.) أى صحيح (True). وإذا لم يجدها يعطى (.F.) أى غير صحيح (False) بينما الدالة (SUBSTR()) تعطى جزءا من السلسلة الحرفية (String).

كما يمكن دراسة المثال التالى لتوضيح طريقة أخرى لاختبار مدخلات المستخدم.

```
name = SPACE(30)
DO WHILE .T.
    @ 5,5 GET name
    READ
    IF name < > SPACE(30)
      EXIT
    ENDIF
    @ 10,10 SAY " No blank name allowed "
ENDDO
```

وهذا البرنامج يستخدم ما يسمى بالطريقة السلبية (Negative Approach) وهو يعنى الخروج من الحلقة التكرارية فى حالة إدخال المستخدم القيمة الصحيحة. فإذا أدخل المستخدم أى حروف فى المتغير (Name) فإن الشرط الموجود بعد (IF) يتحقق وبالتالى يتم تنفيذ الأمر التالى وهو (EXIT) الذى يؤدى إلى الخروج من الحلقة التكرارية وبالتالى تنفيذ باقى أوامر البرنامج.

أما إذا لم يدخل المستخدم أى حروف فإن المتغير (name) يظل خاليا أى محتويا على (SPACE(30)) وذلك يؤدى إلى عدم تحقق الشرط بعد (IF) وبالتالى الإستمرار فى تنفيذ الحلقة التكرارية حتى يقوم المستخدم بإدخال أى حروف فى المتغير (name).

كما يمكن استخدام الطريقة الإيجابية (Positive Approach) وذلك بكتابة الأمر بعد (IF) كالآتى :

```
IF name = SPACE ( 30 )
```

وهناك طرق متعددة لاختبار مدخلات المستخدم ويمكن لمخطط البرامج إختيار الطرق المناسبة ولكن المهم أن يغطى جميع إحتمالات الخطأ.

٣١ - ٣ استخدام الدالة (INKEY())

يمكن استخدام الدالة (INKEY()) فى اختبار مدخلات المستخدم وهذه الدالة تعطى القيمة العددية الممثلة لكود الآسكى (ASCII Code) الخاص بآخر مفتاح تم الضغط عليه بواسطة المستخدم. ولكى يتم توضيح ذلك يمكن دراسة الأوامر التالية :

```
i = o
DO WHILE i = o
   i = INKEY()
ENDDO
? i
```

وهذه الأوامر تؤدى إلى استمرار تنفيذ الحلقة التكرارية حتى يضغط المستخدم على أى مفتاح. فمثلا إذا ضغط المستخدم على المفتاح (A) يترك البرنامج الحلقة التكرارية ويعرض الرقم (65) الذى يمثل كود الآسكى الخاص بالحرف (A).

ويجب ملاحظة أن الدالة (INKEY()) لاتعمل مع مفتاح (ALT) حيث تعطى القيمة صفر وكذلك الضغط على مفتاح (ALT) مع أى مفتاح آخر يؤدى إلى نفس النتيجة.

ويمكن دراسة مجموعة الأوامر التالية للتعرف على استخدام الدالة (INKEY()) فى اختبار مدخلات المستخدم.

```
i = o
DO WHILE i = o
   i = INKEY()
   IF UPPER(CHR(i)) $ "ABCDEFGHIJKLX"
      EXIT
   ENDIF
      i = o
ENDDO
```

وتؤدى هذه الأوامر إلى الدخول فى حلقة تكرارية طالما كانت قيمة المتغير (i) تساوى صفرا. وعندما يضغط المستخدم على أى مفتاح فإن المتغير (i) يحتوى على القيمة العددية الممثلة لكود الآسكى الخاص بهذا المفتاح.

والأمر (IF) يختبر هذه القيمة العددية بعد تحويلها إلى الحرف المقابل باستخدام الدالة (CHR()) وبعد تحويله إلى حرف كبير (CAPITAL). فإذا كان هذا الحرف ضمن الحروف الموجودة فى الشرط يخرج البرنامج من الحلقة التكرارية وينتقل إلى أوامر البرنامج التالية حيث ينفذ الأوامر التى تختص بكل حرف من هذه الحروف حسب اختيار المستخدم.

٣١ - ٤ الضغط على مفتاح الإدخال

فى بعض الأحيان يراد استخدام مفتاح الإدخال كأحد الإختيارات فى البرنامج. فى هذه الحالة يتم اختبار السلسلة الحرفية (String) التى يدخلها المستخدم. فإذا كان طولها = صفر ينفذ البرنامج أوامر معينة. ويمكن توضيح ذلك من الأوامر التالية :

```
IF LEN( TRIM ( choice ) ) = 0
    DO something
ELSE
    DO another
ENDIF
```

حيث أن المتغير (choice) هو متغير يتم إنشاؤه من خلال البرنامج وإعطاؤه القيمة (space) أى أنه سلسلة حرفية خالية.

والشرط بعد (IF) يؤدى إلى اختبار طول السلسلة الحرفية الموجودة فى المتغير (choice) وذلك بعد حذف المسافات الزائدة باستخدام الدالة (TRIM). فإذا ضغط المستخدم على مفتاح الإدخال دون كتابة أى حروف يصبح المتغير (choice) خاليا أى محتويا على فراغ (Space) فقط. كما يؤدى الأمر (TRIM) إلى حذف هذا الفراغ وبالتالى يصبح طول المتغير الحرفى صفرا. وهذا يوضح أهمية الأمر (TRIM) فى هذه الحالة حيث أن السلسلة الحرفية المحتوية على فراغ (space) لايكون طولها صفرا.

٣١ - ٥ إختبار مسطرة المسافات (Space Bar)

كما سبق الإيضاح ، يمكن حذف المسافات الخالية فى أول السلسلة الحرفية أو فى آخرها باستخدام الدوال (LTRIM()) ، (RTRIM()) ، (TRIM()). ولكن ماذا لو أدخل المستخدم مسافة خالية داخل السلسلة الحرفية ؟. فى هذه الحالة لاتصلح هذه الدوال لاكتشاف الخطأ الذى أدخله المستخدم ولكن يمكن استخدام الدالة (AT()). هذه الدالة تختبر وجود حروف معينة داخل سلسلة حرفية فإذا وجدت هذه الحروف فإنها تعطى مكان هذه الحروف وإذا لم تجدها فإنها تعطى القيمة صفر. ولتوضيح ذلك يمكن دراسة الأوامر التالية :

```
DO WHILE .T.
        ACCEPT "What is the account number?" TO choice
        STORE LTRIM(TRIM(choice)) TO choice
        IF AT(" ",choice) > 0
           CLEAR
           ? CHR(7)
              @ 10,10 SAY "You typed space" +;
              "in the account number"
              ?
           WAIT "Press any key to try again"
           RELEASE choice
           LOOP
        ENDIF
        EXIT
 ENDDO (WHILE .T.)
```

وفى هذا البرنامج عندما يدخل المستخدم رقم حساب معين فإن الدالة (LTRIM()) والدالة (TRIM()) تؤديان إلى حذف الفراغات (Spaces) من أول وآخر هذا الرقم ثم تقوم الدالة (AT()) باختبار وجود فراغات داخل هذا الرقم فإذا وجدت أى فراغ فإنها تعطى الرقم الدال على مكان هذا الفراغ فى السلسلة الحرفية (Choice) أى تعطى قيمة أكبر من صفر. وفى هذه الحالة يتحقق الشرط بعد (IF) ويتم تنفيذ الأوامر التالية حيث يتم مسح الشاشة وتشغيل جرس التحذير (Bell) ثم عرض الرسالة

التى توضح للمستخدم الخطأ الذى وقع فيه. كما يؤدى الأمر (WAIT) إلى الإنتظار (Pause) حتى يقرأ المستخدم الرسالة ثم يضغط على أى مفتاح بعد ذلك. ثم يتم مسح محتويات المتغير (choice) باستخدام الأمر (RELEASE). ويلى ذلك الأمر(LOOP) الذى يؤدى إلى الرجوع إلى أول الحلقة التكرارية.

وفى حالة عدم وجود مسافات داخل السلسلة الحرفية بعد (IF) فإن البرنامج يتخطى الاوامر بعد (IF) ثم ينفذ الأمر (EXIT) ليخرج من الحلقة التكرارية بعد تخزين رقم الحساب فى المتغير (choice). ويجب ملاحظة استخدام الأمر (?) دون كتابة أى شيء حيث يؤدى ذلك إلى ترك سطر خال وذلك لعرض الرسالة مع ترك سطر خال بينها وبين الرسالة السابقة.

٣١ - ٦ إختبار نوع المدخلات

هناك طرق أخرى لاختبار مدخلات المستخدم عن طريق اختبار نوع هذه المدخلات إذا كان حرفيا (Character) أو عدديا (Numeric) أو تاريخيا (Date) أو منطقيا (Logical). ويتم ذلك باستخدام الدوال (ISALPHA()) ، (ISLOWER()) ، (ISUPPER()). وهى تعطى القيمة المنطقية صحيح (True) أو غير صحيح (False). وهذه الدوال تختبر أول حرف فقط فى المدخلات ويمكن استخدام هذه الدوال فى اكتشاف أخطاء المستخدم. فمثلا يمكن كتابة البرنامج الفرعى التالى :

```
@ 10,10 GET choice
READ
IF ISALPHA(choice)
     DO something
ELSE
     DO error
ENDIF
```

حيث (error) هو برنامج فرعى آخر يؤدى إلى عرض رسالة للمستخدم توضح له الخطأ بالإضافة إلى تنفيذ بعض العمليات الأخرى.

وهناك دالة أخرى (Type()) تستخدم لتحديد نوع المدخلات وهى لاتعطى قيمة منطقية (صحيح أو غير صحيح) ولكنها تعطى أول حرف يمثل نوع المدخلات مثل

(C) للمدخلات الحرفية (Character) أو (N) للمدخلات العددية (Numeric) أو (D) للمدخلات التاريخية أو (M) للملاحظات (Memo) أو (L) للمدخلات المنطقية (Logical). وفى حالة إدخال حروف غير مطابقة لأى نوع من هذه المدخلات أو عند إدخال حروف فى متغير ذاكرة دون إنشاء هذا المتغير أولا فإن هذه الدالة تعطى الحرف (U) الذى يعنى غير معرف (Undefined) والأوامر التالية توضح استخدام هذه الدالة.

```
STORE "Hello" TO message1
? TYPE('message1')
```

وعند الضغط على مفتاح الإدخال يظهر الحرف (C) أى (Character).

٣١ - ٧ استخدام الأمر (ON)

يستخدم هذا الأمر للتفرع إلى برنامج فرعى آخر بناء على ضغط المستخدم على مفتاح الهروب (ESC) أو أى مفتاح آخر أو عند حدوث خطأ معين فى البرنامج (Error). فمثلا الأمر (ON ESCAPE) يستخدم لاختبار ضغط المستخدم على مفتاح الهروب (ESC) خلال تنفيذ البرنامج فإذا ضغط المستخدم على مفتاح الهروب أثناء تشغيل البرنامج يتفرع البرنامج إلى برنامج فرعى آخر. فمثلا يمكن كتابة الأمر التالى :

```
ON  ESCAPE  DO  warning
```

وهذا يؤدى إلى التفرع إلى برنامج (Warning) عند ضغط المستخدم على مفتاح الهروب. ويمكن كتابة هذا الأمر فى بداية البرنامج الرئيسى حيث يصبح مؤثرا فى جميع البرامج الفرعية.

والأمر (ON KEY) يؤدى نفس الشيء ولكن عند ضغط المستخدم على أى حرف.

أما الأمر (ON ERROR) فإنه يؤدى إلى التفرع إلى برنامج فرعى عند حدوث أى خطأ فى تشغيل البرنامج حيث يمكن من خلال هذا البرنامج الفرعى عرض رسائل خطأ للمستخدم وعدم الإعتماد على رسائل الخطأ المبدئية (Default) الموجودة فى برنامج (DBase III +).

ملاحظة

ما سبق ذكره فى هذا الباب ينطبق أيضا على كل برامج عائلة (DBase) مثل (DBase IV) ، (FoxBase) ، (FoxBase +) ، (FoxPro).

الفصل الثانى والثلاثون

التعامل مع قاعدة البيانات



عند تصميم البرنامج بواسطة أحد برامج عائلة (DBase) فإن هذا البرنامج لايتعامل مع البيانات التى يدخلها المستخدم عن طريق لوحة المفاتيح فقط ولكنه يتعامل أيضا مع البيانات المخزنة فى قاعدة البيانات. وهذا الفصل يشرح تعامل البرنامج مع ملف قاعدة البيانات من حيث تصميم هذا الملف والملفات الملحقة به مثل ملف الفهرس (Index File) أو ملف البحث (Query File) أو ... الخ. وكذلك من حيث فتح هذا الملف والملفات الملحقة به.

٣٢ - ١ تصميم قاعدة البيانات

عادة يتم تصميم قاعدة البيانات خارج البرنامج أى عن طريق برنامج المساعد (Assistant) أو من خلال مشيرة النقطة (Dot Prompt) بأمر منفصل عن أوامر البرنامج. ويستخدم لذلك الأمر (CREATE) ثم كتابة إسم الملف المطلوب إنشاؤه. كما يستخدم الأمر (MODIFY STRUCTURE) فى إنشاء الملف أيضا بالإضافة إلى تعديله.

وملف قاعدة البيانات يستطيع تخزين حتى بليون سجل وكل سجل يحتوى على مايقرب من ٤٠٠٠ حرف موزعين على عدد من الحقول لايزيد عن ١٢٨ حقلا (Field). ولكن يجب ملاحظة أنه عند زيادة حجم ملف قاعدة البيانات بدرجة كبيرة فإن ذلك يؤدى إلى بطء معالجة البيانات المخزنة وبالتالى يؤثر على كفاءة البرنامج.

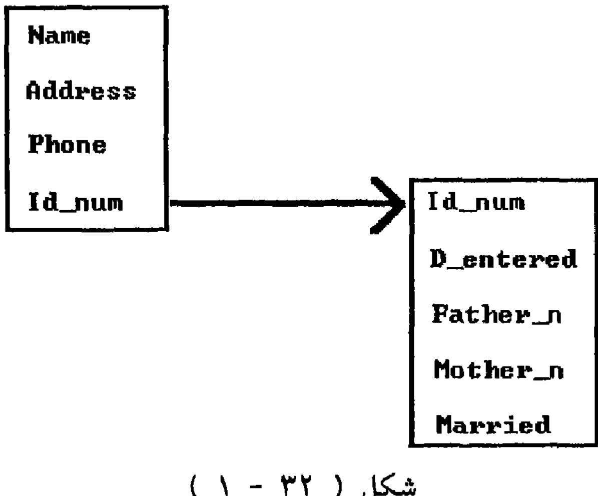
ولعلاج ذلك يمكن تقسيم ملف قاعدة البيانات إلى مجموعة ملفات مع تحديد حقل مشترك بينها مع ملاحظة أن هذا الحقل يجب أن يكون منفردا (Unique) أى يعطى بيانا محددا لكل سجل فى الملف بحيث لايكون هناك سجلان مشتركان فى هذا البيان.

ويسمح برنامج (+DBase III) بتصميم أى عدد من الملفات المرتبطة (Related) دون أى تحديد (Limitation) لهذا العدد ولكن عند فتح هذه الملفات فهناك حد أقصى لعدد الملفات المفتوحة متضمنا ملفات قاعدة البيانات والملفات الملحقة بها مثل الملفات الفهرسية (Index Files) وملفات البحث (Query Files) و ... الخ. وهذا العدد من الملفات يتم تحديده فى ملف المواصفات (Config.sys) ويرتبط بحجم الذاكرة المؤقتة المتاح.

والقاعدة العامة فى استخدام الملفات المتعددة المرتبطة (Related) هى استخدام حقل واحد مشترك ومنفرد (Unique) كما سبق الإيضاح. ويجب تجنب استخدام حقول أخرى

مشتركة بين الملفات حتى لاتؤدى إلى بطء تشغيل البرنامج أو استهلاك المساحة التخزينية المتاحة على القرص.

والشكل التالى يوضح ملفين مرتبطين عن طريق حقل رقم تحقيق الشخصية.



شكل (٣٢ - ١)

ويمكن عن طريق استخدام ملفات متعددة فى قاعدة البيانات عزل البيانات السرية التى يكون مطلوبا عدم التعامل معها بواسطة أشخاص معينين وذلك عن طريق استخدام ملف منفصل لهذه البيانات السرية واستخدام كلمة المرور (Password) فى تحديد الأشخاص المسموح لهم بالتعامل مع هذا الملف. وهذا يعتبر ضروريا عند استخدام البرنامج فى شبكات الحاسب (Networks).

٣٢ - ٢ هيكل ملف قاعدة البيانات

يتم تحديد هيكل ملف قاعدة البيانات باستخدام الأمر (CREATE) أو الأمر (MODIFY STRUCTURE) كما سبق الإيضاح. وفى هذه الحالة يتم فتح هيكل خال للملف لتحديد أسماء الحقول ونوعها وعرضها.

٣٢ - ٢ - ١ تحديد أسماء الحقول

أسماء الحقول (Field Names) يصل طولها إلى عشرة حروف ويجب أن تبدأ بحرف ولاتحتوى على مسافات خالية (Spaces). ويمكن استخدام الحروف والأرقام

والشرطة السفلية (Underscore).

٣٢ - ٢ - ٢ تحديد أنواع الحقول

يتم تحديد أنواع الحقول (Field Types) عن طريق كتابة الحرف الأول من كل نوع. وذلك كالآتى :

C	وتعنى حقل حرفـــى	(أول حرف من Character)
D	وتعنى حقل تاريخـــى	(أول حرف من Date)
L	وتعنى حقل منطقـــى	(أول حرف من Logical)
M	وتعنى حقل ملاحظات	(أول حرف من Memo)
N	وتعنى حقل عـــددى	(أول حرف من Numeric)

كما يمكن تغيير نوع الحقل بالضغط على مسطرة المسافات (Space Bar) حيث تؤدى كل ضغطة عليها إلى التحويل من نوع إلى آخر. وفى الأجزاء التالية يتم شرح كل نوع من الحقول بالتفصيل.

٣٢ - ٢ - ٢ - ١ الحقول الحرفية

وهى حقول يمكن استخدامها فى إدخال أى حروف من لوحة المفاتيح وهى تشمل الحروف والأعداد والحروف الخاصة (Special Characters) والمسافات (Spaces) وأقصى عرض أو حجم لهذه الحقول هو ٢٥٤ حرفا.

٣٢ - ٢ - ٢ - ٢ الحقول التاريخية

وهى حقول تستخدم فى تخزين التواريخ ويمكن عرض التواريخ بعدة صور كما سبق الإيضاح والصورة المبدئية (Default) للتاريخ هى الصورة الأمريكية (mm/dd/yy). وحقل التاريخ دائما عرضه ٨ حروف حتى فى حالة استخدام ٤ حروف ممثلة للسنة (yyyy) باستخدام الأمر (SET CENTURY ON).

ويمكن إجراء عمليات حسابية على هذه الحقول بإضافة عدد من الأيام إلى تاريخ معين أو طرح عدد من الأيام من تاريخ معين أو طرح تاريخ من تاريخ ... وهكذا.

٣٢ - ٢ - ٢ - ٣ الحقول العددية

وهى الحقول التى يتم فيها إدخال الأعداد التى يتم إجراء عمليات حسابية عليها. والحقل العددى يمكن أن يحتوى على ١٥ رقما بما فيها الأرقام العشرية (Decimal) التى يجب ألا تزيد عن ٩ أرقام.

٣٢ - ٢ - ٢ - ٤ الحقول المنطقية

وهى حقول لاتقبل إلا حرفا واحدا يمثل حالة هذا البيان إذا كان صحيحا (True) أو غير صحيح (False). فإذا كان صحيحا يتم إدخال أحد الحروف الآتية (T,t,Y,y) وإذا كان غير صحيح يتم إدخال أحد الحروف الآتية (F,f,N,n).

٣٢ - ٢ - ٢ - ٥ حقول الملاحظات

وهى حقول يتم تصميمها لإدخال كميات كبيرة من المعلومات عن كل سجل. ويتم تخزين هذه المعلومات فى ملف مساعد يحمل نفس إسم ملف قاعدة البيانات ولكن بالإمتداد (dbt.) بدلا من الإمتداد (dbf.). ويتم تمييز هذا الحقل فى ملف قاعدة البيانات بكلمة (memo) وعرضه عشرة حروف.

وحقل الملاحظات حجمه الحقيقى متغير تبعا للمعلومات التى يتم تخزينها فيه ففى حالة عدم إدخال أى بيانات يكون حجمه صفرا. ويمكن إدخال حتى ٥٠٠٠ حرفا فى هذا الحقل. كما يمكن زيادة حجمه عن ذلك كثيرا باستخدام أى برنامج معالجة كلمات آخر غير معالج الكلمات المستخدم فى برنامج (+DBase III). ويلزم لذلك إدخال إسم هذا البرنامج فى ملف المواصفات (Config.sys) الخاص ببرنامج (+DBase III).

٣٢ - ٢ - ٣ تحديد عرض الحقل

عرض الحقل (Field Width) هو أكبر عدد من الحروف والأرقام يمكن كتابته فى الحقل. وفى حالة الحقول العددية يتم حساب نقطة الكسر العشرى

(Decimal Point) والفاصلة (Comma) ضمن حروف الحقل. كما أن الحقول التاريخية والمنطقية وحقول الملاحظات لها عرض ثابت.

٣٢ - ٢ - ٤ فتح ملف قاعدة البيانات

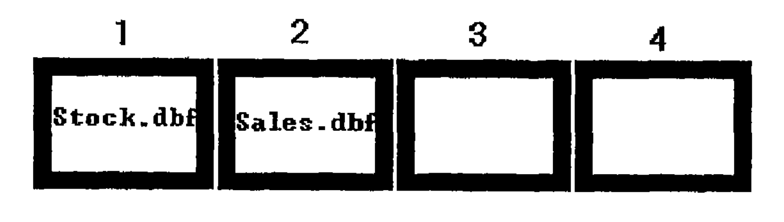
يتم فتح ملف قاعدة البيانات باستخدام الأمر (USE) يليه إسم الملف المطلوب فتحه. ويمكن فتح ملفات قاعدة البيانات وملفات الفهرس الملحقة بها فى بداية البرنامج كما يمكن فتح الملفات المطلوبة وقت الحاجة اليها وهذا يتوقف على حجم البرنامج وعدد الملفات المستخدمة. ولكن يفضل فى جميع الأحوال فتحها وقت الإحتياج اليها فقط للمحافظة على تكامل قاعدة البيانات (Integrity) حيث أن ترك الملفات مفتوحة مدة طويلة قد يؤدى إلى حدوث مشاكل فى هذه الملفات فى حالة قطع التيار الكهربى أو الإغلاق المفاجئ للجهاز. كما يفضل إغلاق الملف بمجرد انتهاء الحاجة اليه.

وفى حالة استخدام عدة ملفات مرتبطة (Related) فى نفس الوقت يتم فتح هذه الملفات فى مناطق عمل (Work Areas) مختلفة باستخدام الأمر(SELECT).

فمثلا مجموعة الأوامر التالية تؤدى إلى فتح ملفين فى منطقتين للعمل(2 , 1).

```
SELECT 1
USE Stock INDEX Cust_no
SELECT 2
USE Sales INDEX Sale_no
```

ويلاحظ فتح كل ملف بالاضافة إلى فتح الملف الفهرسى المرتبط به من خلال نفس الأمر. والشكل التالى يوضح عملية تخصيص منطقة العمل (Work Area) لكل ملف.



شكل (٣٢ - ٢)

ويمكن فتح حتى عشرة مناطق عمل (Work Areas) واستخدامها فى فتح عدة ملفات فى نفس الوقت. ويمكن إغلاق أى ملف وبالتالى إغلاق منطقة العمل الخاصة به وذلك باستخدام الأمر (USE) دون كتابة أى شىء بعده كما يمكن إغلاق جميع ملفات قاعدة البيانات باستخدام الأمر (CLOSE DATABASES).

٣٢ - ٣ استخدام المرادفات (Aliases)

عندما يراد استخدام عدة ملفات قاعدة بيانات من خلال مناطق عمل مختلفة (Work Areas) يمكن اختيار مرادفات (Aliases) لإسم كل ملف. وهذه المرادفات توفر على مخطط البرامج كتابة إسم الملف وملف الفهرس (Index File) الملحق به فى كل مرة يراد فيها اختيار ملف فى منطقة عمل (Work Area) معينة.

فمثلا عند كتابة الأوامر التالية :

```
SELECT 1
USE Cadets Index Name Alias Cadets
```

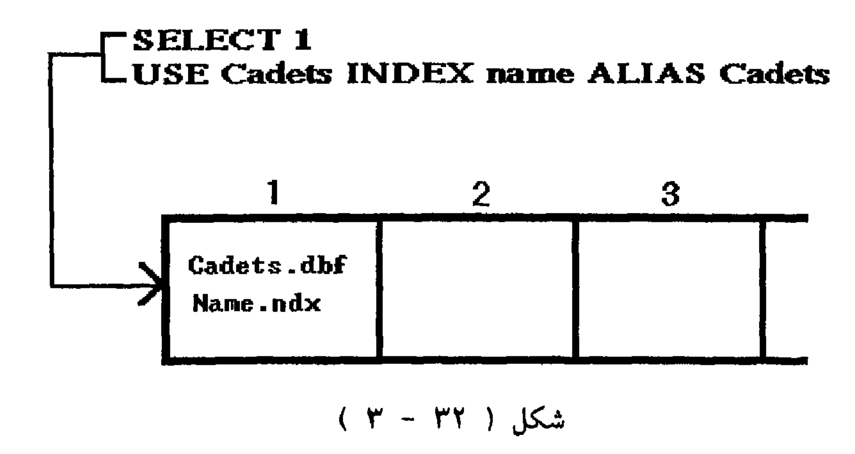
وفى كل مرة يراد فتح ملف الطلبة (Cadets) وملف الفهرس الملحق به يستخدم الأمر التالى :

```
SELECT Cadets
```

وهذه المرادفات (Aliases) تساعد على توضيح أى قاعدة بيانات مفتوحة فى أى منطقة عمل.

تحذير

يجب عدم استخدام الحروف من (A) إلى (J) منفصلة فى المرادفات (Aliases) لأن هذه الحروف تمثل مناطق العمل من ١ الى ١٠. ولكن يمكن استخدام هذه الحروف مع حروف أخرى أو أعداد أخرى مثل (A1) ، (A2) ، (A3) ، ... الخ.



شكل (٣٢ - ٣)

٣٢ - ٤ إنشاء ملف الفهرس (Index File)

كما سبق الإيضاح فإن الملف الفهرسى (Index File) هو ملف يحتوى على حقلين فقط أحدهما يحتوى على رقم السجل والآخر يحتوى على البيانات المطلوب ترتيب ملف قاعدة البيانات بناء عليها. وعن طريق هذا الملف يمكن الوصول بسهولة إلى أى سجل فى ملف قاعدة البيانات عن طريق البيانات الموجودة فى الحقل الفهرسى (Key Field). ولإنشاء ملف الفهرس يستخدم الأمر (INDEX) مع كتابة إسم الحقل الفهرسى وكذلك إسم الملف المطلوب إنشاؤه. وذلك كالآتى مثلا :

```
INDEX ON name TO Name
```

وهذا يعنى إنشاء ملف فهرسى إسمه (Name.ndx) بناء على حقل الإسم (name) كحقل فهرسى.

ويمكن الترتيب بناء على أكثر من حقل. وهذا يساعد على ترتيب السجلات التى تشترك فى الحقل الفهرسى الأول بناء على حقل آخر. فمثلا إذا كان هناك حقل يمثل الإسم الأول (First Name) وحقل يمثل الإسم الثانى (Second Name) فإن البرنامج يبدأ بترتيب قاعدة البيانات بناء على حقل الإسم الأول. فإذا كان هناك عدة سجلات تشترك فى الإسم الأول وليكن (Mohamed) مثلا وذلك كالآتى :

Mohamed Maged
Mohamed Hasan
Mohamed Tarek

فى هذه الحالة يتم ترتيب هذه السجلات بناء على الإسم الثانى (Second Name) وذلك كالآتى :

MOHAMED Hasan
Mohamed Maged
Mohamed Tarek

ويلاحظ هنا إعادة ترتيب السجلات بناء على الترتيب الهجائى للإسم الثانى.

ولكى يتم إنشاء ملف فهرسى (Index File) بناء على حقلين أو أكثر تستخدم علامة الجمع (+) وذلك كالآتى :

```
USE Cadets
INDEX ON F_name + L_name TO Name
```

ويجب ملاحظة أن الترتيب يتم تصاعديا حسب الترتيب الهجائى للحروف.

وعند الترتيب بناء على عدة حقول يجب التأكد أولا أن هذه الحقول لها نفس النوع (Type). وإذا كان النوع مختلفا فيلزم فى هذه الحالة إجراء عمليات تحويل من حالة إلى أخرى.

فمثلا إذا أريد ترتيب بيانات ملف الطلبة (Cadets) بناء على حقل الإسم الأول (F_name) وحقل الإسم الثانى (S_name) وحقل تاريخ الدخول (D_enter) فى هذه الحالة يلزم تحويل حقل التاريخ إلى حروف. حيث يتم عمل تحويل منفصل للجزء الممثل للسنة والجزء الممثل للشهر والجزء الممثل لليوم وذلك كالآتى مثلا :

```
USE Cadets
INDEX ON F_name + S_name + STR(YEAR(D_Enter),4) ; +
STR(MONTH(D_Enter),2) + STR(DAY(D_Enter),2)
```

٣٢ - ٥ فتح ملف الفهرس

يتم فتح ملف الفهرس (Index File) باستخدام نفس الأمر (USE) المستخدم فى فتح ملف قاعدة البيانات. وذلك كالآتى مثلا :

USE Cadets INDEX Name

وهناك طريقة أخرى لفتح ملف الفهرس باستخدام الأمر (SET INDEX TO) ثم كتابة إسم الملف الفهرسى. وذلك كالآتى :

SET INDEX TO Name, Class

ويجدر العلم أنه عند فتح ملفات الفهرس فإن أى تعديل أو إضافة للبيانات تؤدى إلى تعديل ملف الفهرس أيضا. أى أن أى سجلات تضاف إلى قاعدة البيانات يتم إدخالها فى نفس الوقت فى ملف الفهرس.

ويمكن تغيير ترتيب ملفات الفهرس التى سبق فتحها باستخدام الأمر (SET ORDER TO) ثم كتابة رقم ملف الفهرس المراد إستخدامه كملف فهرسى رئيسى (Master). وذلك كالآتى مثلا :

SET ORDER TO 2

فى هذه الحالة يصبح الملف (Class) هو ملف الفهرس الرئيسى (Master).

ويمكن إعادة الملف إلى الحالة غير المفهرسة (Unindexed) وذلك باستخدام الأمر (SET ORDER TO 0). وهذا يؤدى إلى التعامل مع السجلات بترتيبها الأصلى الذى أدخلت به وذلك مع عدم إغلاق ملفات الفهرس.

ولإغلاق ملفات الفهرس دون إغلاق ملف قاعدة البيانات يستخدم الأمر التالى :

CLOSE INDEX

كما يمكن إغلاقها أيضا باستخدام الأمر التالى :

SET INDEX TO

وذلك دون كتابة أى شيء بعده.

ملاحـظة

رغم أن عدد ملفات الفهرس التى يمكن إنشاؤها لملف قاعدة بيانات واحد غير محدود حيث يمكن إنشاء أى عدد من ملفات الفهرس لكل ملف قاعدة بيانات إلا أن عدد ملفات الفهرس التى يمكن فتحها فى نفس الوقت لايزيد عن سبعة ملفات. وهذا يعتبر كافيا جدا إذا أخذنا فى الإعتبار أيضا أن الحقل الفهرسى فى كل ملف يمكن أن يحتوى على أكثر من حقل من حقول ملف قاعدة البيانات.

ويفضل دائما فتح جميع ملفات الفهرس المطلوبة مرة واحدة ثم تعديل ترتيب هذه الملفات فى أى وقت باستخدام الأمر (SET ORDER TO) وذلك حتى يتم تحديث جميع ملفات الفهرس مع أى تعديل لبيانات ملف قاعدة البيانات.

٣٢ - ٦ البحث عن سجل معين

يمكن أن نتخيل عملية البحث عن سجل معين فى الملف كأن هناك مؤشرا معينا يتحرك على أرقام السجلات ليقف عند رقم معين. وطالما كان هذا المؤشر موجودا على هذا الرقم فإن أى عرض للسجلات باستخدام أوامر عرض البيانات المعروفة يؤدى إلى عرض بيانات هذا السجل فقط. فمثلا عند كتابة الأمر (GOTO) وبعده رقم السجل المطلوب فإن المؤشر يتحرك حتى يصل إلى هذا السجل. وعند استخدام الأمر(DISPLAY) مثلا يلاحظ ظهور بيانات هذا السجل.

ويمكن من خلال البرنامج الوصول إلى السجلات التى يطلبها المستخدم وذلك عن طريق سؤاله عن رقم السجل الذى يريده ثم الذهاب إلى هذا السجل وعرض بياناته. وذلك كالآتى مثلا :

```
GOTO 5
DISPLAY
```

ولكن ماذا لو كان المستخدم غير متذكر لرقم السجل الذى يريده ؟ وهذا هو ما يحدث فى الغالب. فى هذه الحالة فإن البرنامج يجب أن يتيح للمستخدم الوصول إلى السجل أو السجلات المطلوبة عن طريق مطابقة بيانات حقل معين فى هذا السجل لقيمة محددة يدخلها المستخدم إلى البرنامج. ويقوم البرنامج بعد ذلك بتوجيه المؤشر إلى هذا السجل حتى يسهل بعد ذلك عرض بياناته باستخدام أى أمر من أوامر عرض البيانات مثل (?,SAY, DISPLAY, LIST ... @).

وعند استخدام ملف الفهرس فإن عملية البحث تكون سريعة خصوصا إذا كان الحقل المطلوب البحث بواسطته هو الحقل الفهرسى (Key Field). وهناك عدة أوامر تستخدم فى البحث عن السجلات بعضها يشترط فتح ملف الفهرس أولا وبعضها الآخر لايشترط ذلك. وسيتم دراستها بالتفصيل فى الأجزاء التالية.

٣٢ - ٦ - ١ استخدام الأمر (LOCATE)

يستخدم هذا الأمر فى البحث عن سجل معين فى قاعدة البيانات سواء كانت مفهرسة (Indexed) أو غير مفهرسة (Unindexed). ويتم ذلك عن طريق إختبار تحقيق هذا السجل لشرط معين. وهذا الشرط يكون عبارة عن علاقة بين البيانات الموجودة فى حقل معين وبين قيمة محددة. وعند تحقيق أى سجل أو مجموعة من السجلات لهذا الشرط فإن المؤشر يقف عند أول سجل يحقق هذا الشرط. ويمكن التعامل مع البيانات الموجودة فى هذا السجل حسب الحاجة.

وعندما يراد الوصول إلى سجل آخر يحقق الشرط يستخدم الأمر(CONTINUE) وهو لايستخدم إلا مع الأمر (LOCATE). فمثلا للوصول إلى السجل الخاص برقم الحساب (60789) يستخدم الأمر التالى :

```
LOCATE FOR Acct_no = '60789'
```

والأمر (LOCATE) هو أبطأ أوامر البحث لأنه لايعتمد على ملف الفهرس (Index File) ولذلك فهو يبحث فى كل الملف حتى يجد الحقل الذى يحقق الشرط.

٣٢ - ٦ - ٢ الأمر (FIND) والأمر (SEEK)

هذان الأمران فى منتهى القوة لأنهما يؤديان إلى الوصول إلى السجل المطلوب بسرعة كبيرة جدا مهما كان حجم ملف قاعدة البيانات المستخدم ولكن استخدامهما يجب أن يتم بمنتهى الحذر ويتطلب ذلك دراسة كل منهما والتعرف على خصائصه. فمن خصائصهما مثلا أنهما لايعملان إلا على ملف مفهرس (Indexed) وأن يكون الحقل الفهرسى (Key Field) هو الحقل الذى يتم البحث عن طريقه.

فمثلا عند البحث عن إسم معين فى قاعدة بيانات الطلبة (Cadets) وليكن (SHEREIF) مثلا يتم كتابة الأمر التالى :

```
FIND SHEREIF
```

وهذا يساوى تماما الأمر

```
SEEK " SHEREIF "
```

ويلاحظ هنا ضرورة وضع السلسلة الحرفية (String) بين علامات تنصيص (Quotation) فى حالة استخدام الأمر (SEEK) فقط. ولكن ليس هناك حاجة لاستخدام علامات التنصيص مع الأمر (FIND).

ويتم البحث عن الإسم عن طريق مقارنة كل حرف فى هذا الإسم بالحرف المقابل فى بيانات الحقل الفهرسى بدءا من أول حرف والإستمرار حرفا حرفا حتى الوصول إلى نهاية الإسم المطلوب البحث عنه. فمثلا عند استخدام الأمر السابق يقف المؤشر عند أول سجل يحتوى حقل الإسم فيه على الإسم (SHEREIF). فإذا كان هناك حقل يحتوى على الإسم (SHEREIFA) مثلا فإن المؤشر يقف عنده رغم أن المطلوب هو الاسم (SHEREIF) وليس (SHEREIFA).

واذا أريد الوصول إلى السجل المطابق تماما للإسم المطلوب يتم استخدام الأمر (SET EXACT ON) كما سيتم الإيضاح فيمابعد. ولكن بدون استخدام هذا الأمر فإن البحث عن سلسلة حرفية معينة أو حرف معين يعنى البحث عن أى حقل يبدأ بهذه السلسلة أو هذا الحرف.

فمثلا للوصول إلى أول سجل يبدأ الإسم فيه بالحرف (A) يتم كتابة الأمر التالى :

```
SEEK "A"
```

والأمر (SEEK) أقوى وأكثر شمولا من الأمر (FIND) حيث أن الأمر (FIND) يتعامل مع المدخلات كحروف فقط حتى إذا أريد البحث عن عدد معين فإنه يتعامل مع هذا العدد كسلسلة حرفية. فمثلا عند كتابة الأمر التالى :

```
FIND 125.60
```

فإن البرنامج يقارن جميع مدخلات الحقل الفهرسى حتى يصل إلى الرقم المطابق تماما لهذا العدد.

أما الأمر (SEEK) فإنه يتعامل مع المدخلات الحرفية والعددية والتاريخية بالإضافة إلى أى علاقات (Expressions). لذلك فمن الضرورى لإدخال قيمة حرفية كتابتها بين علامات تنصيص كما سبق الإيضاح.

وللبحث عن تاريخ معين مثلا يتم كتابة الأمر التالى :

```
SEEK  CTOD ( ' 05/01/90 ' )
```

مع ملاحظة فتح الملف الفهرسى الذى يستخدم حقل التاريخ كحقل فهرسى رئيسى (Master Index).

واستخدام هذين الأمرين (FIND) و (SEEK) يؤدى إلى تحريك المؤشر إلى أول ملف قاعدة البيانات بصرف النظر عن مكان هذا المؤشر قبل استخدام الأمر. وإذا لم يجد البرنامج القيمة المطابقة فإن المؤشر يتحرك إلى نهاية ملف قاعدة البيانات. كما تظهر الرسالة التالية :

```
No Find
```

ويمكن استخدام هذين الأمرين مع متغيرات الذاكرة مع ملاحظة استخدام التعويض بالماكرو (Macro Substitution) عند استخدام الأمر(FIND) مع متغيرات الذاكرة. فمثلا عند إدخال الإسم (Mohamed) فى متغير الذاكرة (mname) كالآتى :

```
INPUT "Mohamed" TO mname
```

فعند استخدام الأمر (SEEK) فى البحث عن محتويات المتغير (mname) فى ملف قاعدة البيانات يكتب الآتى :

```
SEEK mname
```

أما عند استخدام الأمر (FIND) لتنفيذ نفس الشيء فيكتب كالآتى :

```
FIND & mname
```

ويلاحظ هنا إستخدام أمر التعويض بالماكرو (&) وهذا الأمر سيتم شرحه بالتفصيل فيما بعد. والمثال التالى يوضح استخدام الأمر (FIND) من خلال برنامج فرعى (Module).

```
SET TALK OFF
CLEAR
m_name = SPACE(20)
@ 5,5 SAY 'Enter a name'
@ 5,30 GET m_name PICTURE "@A"
READ
STORE LTRIM(TRIM(m_name)) TO m_name
USE Cadets INDEX name
FIND & m_name
CLEAR
@ 1,5 SAY TRIM(name)
@ 3,5 SAY address
RELEASE m_name
RETURN
```

والسطر الأول من البرنامج يؤدى إلى عدم ظهور خطوات تنفيذ البرنامج على الشاشة كما سبق الإيضاح.

والسطر الثانى يؤدى إلى مسح الشاشة.

والسطر الثالث يؤدى إلى إنشاء متغير ذاكرة (m_name) طوله عشرون حرفا.

والسطر الرابع يؤدى إلى عرض رسالة للمستخدم لتوضيح المطلوب.

والسطر الخامس يؤدى إلى ظهور عمود ضوئى بطول عشرين حرفا حتى يقوم المستخدم بإدخال الإسم المطلوب فيه.

والسطر السادس يؤدى إلى تخزين الإسم الذى يدخله المستخدم فى متغير الذاكرة (m_name).

والسطر السابع يؤدى إلى التخلص من المسافات الموجودة فى أول الإسم أو فى آخره.

والسطر الثامن يؤدى إلى فتح ملف قاعدة البيانات (Cadets) وملف الفهرس المرتبط به (name).

والسطر التاسع يؤدى إلى البحث عن الإسم الذى يدخله المستخدم ويلاحظ هنا إستخدام أمر التعويض بالماكرو (Macro Substitution).

والسطر العاشر يؤدى إلى مسح الشاشة تمهيدا لعرض البيانات الخاصة بالسجل الذى يتم الوصول اليه.

والسطران الحادى عشر والثانى عشر يؤديان إلى عرض الإسم والعنوان الخاص بهذا السجل.

والسطر الثالث عشر يؤدى إلى إلغاء متغير الذاكرة (m_name) حتى يستطيع المستخدم إدخال إسم آخر للبحث عنه.

والسطر الأخير يؤدى إلى العودة إلى البرنامج الرئيسى الذى قام باستدعاء هذا البرنامج الفرعى.

ويجب ملاحظة أن هذا البرنامج يصل فقط إلى أول سجل يحتوى على الإسم الذى يكتبه المستخدم. وبمجرد وقوف المؤشر عند هذا السجل يتم عرض الحقول المطلوبة من هذا السجل باستخدام الأمر (@...SAY) كما يتضح من البرنامج.

٣٢ - ٦ - ٣ عرض بيانات جميع السجلات التى تحقق الشرط

كما سبق الإيضاح فى البرنامج السابق فإن الأمرين (SEEK) و (FIND) يؤديان إلى وضع المؤشر عند أول سجل يحقق الشرط وبالتالى يمكن عرض بيانات هذا السجل.

ولكن ماذا لو أراد المستخدم عرض جميع السجلات التى تحقق الشرط ؟ وذلك كما يحدث مع الأمر (LOCATE) باستخدام الأمر (CONTINUE). فى هذه الحالة يلزم استخدام وسيلة معينة لتحريك المؤشر من أول سجل يحقق الشرط إلى السجل الثانى والثالث و ... وهكذا. ويتم ذلك عن طريق استخدام الحلقة التكرارية باستخدام الأمر (DO WHILE).

فمثلا البرنامج التالى تعديل للبرنامج السابق بحيث يحقق المطلوب :

```
SET TALK OFF
CLEAR
m_name = SPACE(20)
@ 5,5 SAY 'Enter a name'
@ 5,30 GET m_name PICTURE "@A"
READ
STORE LTRIM(TRIM(m_name)) TO m_name
USE Cadets INDEX name
FIND & m_name
CLEAR
r = 1
```

```
DO WHILE TRIM(name) = m_name
     @ r,10 SAY TRIM(name)
     @ r+1,10 SAY address
     ?
     SKIP
     r = r+4
ENDDO WHILE TRIM(name) = m_name
RELEASE m_name
RETURN
```

ويلاحظ أن الجزء الأول من البرنامج مطابق للبرنامج السابق تماما ثم تمت إضافة الحلقة التكرارية التى تبدأ بالأمر (DO WHILE). وقد تم إنشاء متغير ذاكرة (r) قبل بداية الحلقة التكرارية لاستخدامه فى تحديد السطر الذى يتم استخدامه فى عرض البيانات على الشاشة.

والجزء الأول من البرنامج يؤدى إلى الوصول إلى أول سجل يحتوى على الإسم الذى يكتبه المستخدم كما سبق الإيضاح من البرنامج السابق. أما الجزء الخاص بالحلقة التكرارية فإنه يؤدى إلى تحريك المؤشر إلى باقى السجلات التى تحقق الشرط وعرض بياناتها واحدا تلو الآخر. ويتم ذلك عن طريق استخدام هذا الشرط فى أول الحلقة التكرارية. أى أن الحلقة التكرارية تبدأ فى عرض بيانات الإسم والعنوان للسجل الثانى الذى يحقق الشرط ثم عن طريق الأمر (SKIP) يتم نقل المؤشر إلى السجل التالى ثم تتكرر الحلقة التكرارية. وفى كل مرة يتم اختبار الشرط فى أول الحلقة وبالتالى لايتم عرض بيانات الإسم والعنوان إلا للسجلات التى تحقق الشرط.

ويلاحظ هنا إستخدام الأمر :

```
r = r + 4
```

وذلك لزيادة عدد السطور وبالتالى ظهور بيانات السجلات وبينها سطور خالية. ويجب ملاحظة أن أى كلمات مكتوبة بعد الأمر (ENDDO) هى ملاحظات تظهر فقط لمخطط البرامج عند عرض الأوامر على الشاشة ولكنها لاتؤثر على تشغيل البرنامج. وهى تفيد فى توضيح أى أمر (DO WHILE) يتبع له هذا الأمر

(ENDDO). وهذا يكون ضروريا بصفة خاصة فى البرامج التى تحتوى على عدة حلقات تكرارية حتى يكون التسلسل المنطقى للبرنامج واضحا. وهذا البرنامج يؤدى إلى عرض جميع السجلات التى تحقق الشرط على الشاشة.

ملاحظة

عندما يزيد عدد السجلات عن عدد سطور الشاشة أى يصبح (r) أكبر من (٢٤) فى هذه الحالة يتوقف ظهور باقى السجلات وتظهر رسالة خطأ. لذلك يلزم إضافة أوامر أخرى للبرنامج تؤدى إلى ظهور شاشة جديدة عند الوصول إلى آخر سطر فى الشاشة وهذا سيتم شرحه فيما بعد.

٣٢ - ٧ إختبار نهاية الملف

عند استخدام أى أمر من أوامر البحث مثل (SEEK) أو (FIND) أو (LOCATE) فمن المهم اختبار نهاية الملف (End of File) وذلك لأن البرنامج عند بحثه عن قيمة معينة سواء كانت حرفية أو عددية فإنه يبدأ البحث من أول سجل فى الملف ويستمر فى فحص السجلات واحدا بعد الآخر. فإذا لم يجد أى سجل يحقق الشرط فإن المؤشر يصل إلى نهاية الملف. وقد يسبب هذا الحصول على نتائج غير سليمة خصوصا عند استخدام بعض الأوامر التى تبدأ البحث من المكان الذى يقف عنده المؤشر. لذلك فإن برنامج (+DBase III) يتيح لمخطط البرامج اختبار نهاية الملف عن طريق دالة خاصة تسمى (EOF()). هذه الدالة تعطى القيمة صحيح (.T.) أى (True) أو غير صحيح (.F.) أى (False) بناء على وضع المؤشر إذا كان فى نهاية الملف أو ليس فى نهاية الملف.

ملاحظة

هناك فرق بين نهاية الملف (End of File) وبين آخر سجل فى قاعدة البيانات (BOTTOM) حيث أن نهاية الملف تكون نقطة بعد آخر سجل مباشرة. وكذلك فإن بداية الملف تختلف عن أول سجل فى الملف (TOP). وعند كتابة برنامج مثل البرنامج السابق يجب إضافة مجموعة من الأوامر التى تؤدى إلى اختبار الوصول إلى نهاية الملف.

ولتوضيح ذلك يمكن كتابة نفس البرنامج السابق كالآتى :

```
SET TALK OFF
m_name = SPACE(20)
@ 5,5 SAY 'Enter a name'
@ 5,30 GET m_name PICTURE "@A"
READ
STORE LTRIM(TRM(m_name) TO m_name
USE Cadets INDEX name
FIND & m_name
   IF EOF()
      CLEAR
      @ 5,1 SAY 'Sorry, there is no' + m_name
      @ 15,1 WAIT
   ELSE
      CLEAR
      r = 1
      DO WHILE TRIM (name) = m_name.AND..NOT.EOF()
            @ r,10 SAY TRIM(name)
            @ r+1,10 SAY address
            ?
            SKIP
             r = r + 4
      ENDDO WHILE TRIM(name) = m_name
    ENDIF
RELEASE m_name
RETURN
```

ويلاحظ هنا أن هناك نقطتين يتم عندهما إختبار نهاية الملف النقطة الأولى فى السطر التالى للأمر (FIND) مباشرة باستخدام الأمر (IF) والنقطة الثانية فى بداية الحلقة التكرارية عندما ينتقل المؤشر إلى سجل جديد باستخدام الأمر (SKIP) حيث يتم اختبار نهاية الملف عند الرجوع إلى أول الحلقة التكرارية (DO WHILE).

٣٢ - ٨ إستخدام دالـــة رقم السجل

تستخدم الدالة (RECNO()) لتحديد رقم السجل الذى يقف عنده المؤشر ويمكن تخزين هذا الرقم فى متغير ذاكرة واستخدامه فيمابعد وذلك كالآتى مثلا :

```
FIND & m_name
IF .NOT. EOF()
   STORE RECNO() TO record_n
ENDIF
```

فى هذه الحالة يتم البحث عن محتويات المتغير (m_name) فى ملف قاعدة البيانات. فإذا وجد البرنامج سجلا يحتوى على هذه المحتويات فإنه يخزن رقم هذا السجل فى متغير الذاكرة (record_n) وإذا لم يجده يصل إلى نهاية الملف. ويمكن بعد ذلك الذهاب إلى السجل الذى تم تخزين رقمه فى المثال السابق باستخدام الأمر (GOTO) كالآتى :

```
GOTO record_n
```

كما يمكن عرض بيانات هذا السجل أو مسحه حسب الحاجة.

٣٢ - ٩ إستخدام الدالة (FOUND())

فى بعض الأحيان يكون مطلوبا فقط مجرد معرفة إذا كان السجل المطابق موجودا أم لا وليس مهما الوصول إلى هذا السجل أو عرض بياناته. فمثلا عندما يقوم المستخدم بإدخال حساب معين فإن البرنامج يجب أن يتأكد أن هذا الرقم لم يسبق إدخاله حتى لايتم إدخال سجلات مكررة وفى هذه الحالة يمكن كتابة السطور التالية :

```
SEEK macct
IF FOUND()
   @ 10,10 SAY "This number already exists"
ELSE
   EXIT
ENDIF
```

والدالة (FOUND()) فى هذا المثال تعطى القيمة صحيح (True) أو غير صحيح (False). فإذا أعطت القيمة صحيح فإن ذلك يعنى أن الأمر (SEEK) قد وجد رقم حساب يطابق الرقم الذى تم إدخاله فى المتغير (macct). وفى هذه الحالة تظهر الرسالة المبينة للمستخدم لتحذره من إدخال هذا الرقم. ويلاحظ هنا أنه ليس مهما تحديد السجل المطابق للرقم ولكن المهم معرفة إذا كان هناك سجل مطابق أم لا.

٣٢ - ١٠ استخدام المرشح (Filter)

يمكن استخدام المرشحات (Filters) كوسيلة أخرى لتجميع السجلات التى تحقق شرطا معينا وذلك عندما يراد إجراء عملية معينة على هذه السجلات مثلا مثل تجميع الأعداد الموجودة فى حقل معين. ويستخدم لإنشاء هذه المرشحات الأمر (SET FILTER TO) ثم كتابة الشرط أو الشروط المطلوب استخدامها وذلك كالآتى مثلا :

```
SET FILTER TO  M $ name  .AND.  age < 30
```

والشرط الأول (M $ name) يعنى كل الأسماء التى تحتوى على الحرف (M) والشرط الثانى يعنى الأشخاص الذين تقل أعمارهم عن ٣٠ سنة. وهذا السطر يؤدى إلى تصفية قاعدة البيانات وعدم السماح بالمرور من المرشح إلا للأسماء التى تحقق الشرطين معا أى أسماء الأشخاص الذين تحتوى أسماؤهم على الحرف (M) وتقل أعمارهم عن ٣٠ سنة.

ملاحظة

يراعى نقل المؤشر إلى أول الملف بعد استخدام الأمر (SET FILTER TO) وقبل إجراء أى عملية أخرى وذلك لأن هذا الأمر مثل الأمر (SEEK) أو الأمر (FIND) يقوم بالمرور على جميع السجلات وبالتالى فإنه ينقل المؤشر إلى آخر الملف. ولذلك فعند إجراء أى عملية مثل تجميع حقل عددى مثلا فإن التجميع يبدأ من مكان المؤشر وبالتالى لاتكون النتيجة سليمة. ولذلك يستخدم الأمر (GO TOP) بعد استخدام المرشح مباشرة للوصول إلى بداية الملف مرة أخرى.

٣٢ - ١١ إستخدام الدالــة (DELETED())

عندما يراد مسح مجموعة من السجلات فإن برنامج (+DBase III) يتيح لمخطط البرامج مسح السجلات على مرحلتين. المرحلة الأولى يتم خلالها وضع علامات على السجلات المطلوب مسحها والمرحلة الثانية يتم خلالها تنفيذ عملية المسح. ويتيح ذلك للمستخدم التأكد أنه يريد مسح هذه السجلات قبل المسح الفعلى لها. كما يتيح له أيضا الاحتفاظ بالسجلات المطلوب مسحها أطول فترة ممكنة قبل مسحها فعليا من قاعدة البيانات. وفى هذه الحالة يجب عزل هذه السجلات حتى لاتؤثر فى سرعة البحث عن أى سجل وحتى يمكن إجراء أى عمليات على السجلات المطلوبة فقط دون إضاعة الوقت فى تنفيذها على سجلات مطلوب مسحها.

ويتم عزل السجلات المطلوب مسحها بطريقتين :

الطريقة الأولى باستخدام الأمر (SET DELETED ON) حيث أن الوضع المبدئى (Default) لهذا الأمر يكون (OFF).
والطريقة الثانية باستخدام الدالة (DELETED()) كالآتى :

```
SET FILTER TO .NOT. DELETED()
```

وهذا يؤدى إلى عزل السجلات التى سبق تجهيزها للمسح.

٣٢ - ١٢ استخدام الأمر (SET EXACT ON)

عندما يقارن برنامج (+DBase III) بين سلسلتين حرفيتين (Strings) فإنه يقارن كل حرف من السلسلة الأولى بالحرف المقابل له من السلسلة الثانية. وتستمر هذه المقارنة حتى تنتهى السلسلة اليمنى. فمثلا عند مقارنة متغير يحتوى على الإسم ("Mohamed") بمتغير آخر يحتوى على الحروف ("Moh") يتم كتابة السطر التالى :

```
? " Mohamed "  =  " Moh "
```

وعند الضغط على مفتاح الإدخال تظهر القيمة (.T.) أى أن المقارنة صحيحة وذلك بالرغم من عدم تطابق الطرفين. وذلك لأن البرنامج يقارن أولا الحرف الأول فى الطرفين فيجده

مطابقا فينتقل إلى الحرف الثانى فيجده مطابقا ثم ينتقل إلى الحرف الثالث فيجده مطابقا ثم لايجد حروفا أخرى فى الطرف الأيمن لذلك يعطى القيمة (.T.) أى صحيح.

وإذا استبدلنا الطرفين فى السطر السابق أى تمت كتابته كالآتى مثلا :

? " Moh " = " Mohamed "

فإن النتيجة فى هذه الحالة تصبح العكس وذلك لأن الطرف الأيسر ينتهى قبل الأيمن. ولذلك يعطى البرنامج القيمة (.F.) أى غير صحيح (False).

وعند استخدام الأمر (SEEK) أو الأمر (FIND) يحدث نفس الشئ. فمثلا عند البحث عن الحروف (Moh) فى حقل الإسم يستخدم الأمر التالى :

SEEK " Moh "

وفى هذه الحالة يقارن البرنامج بين كل إسم موجود فى الحقل وبين الحروف ("Moh") ثم يقف عند أول إسم يبدأ بهذه الحروف. وهذا قد يكون مطلوبا فى بعض الأحوال عندما يراد مثلا البحث عن الإسم الذى يبدأ بهذه الحروف. ولكن فى أحوال أخرى قد يكون مطلوبا البحث عن إسم محدد وفى هذه الحالة يستخدم الأمر (SET EXACT ON) وهذا الأمر يساعد على البحث عن سلسلة حرفية معينة حتى يصل البرنامج إلى سلسلة حرفية مطابقة لها تماما فى الملف.

ملاحظة

يراعى عند استخدام هذا الأمر (SET EXACT ON) أن يتم استخدامه فى البحث المطلوب ثم إعادته إلى الوضع المبدئى (Default) مرة أخرى بمجرد الإنتهاء من البحث وذلك عن طريق الأمر (SET EXACT OFF).

٣٢ - ١٣ منع الازدواج (Duplication)

فى بعض قواعد البيانات تكون هناك بعض الحقول المنفردة (Unique) أى التى تحتوى على قيمة مختلفة لكل سجل من سجلات الملف. فمثلا فى قواعد البيانات الخاصة بالحسابات يكون هناك رقم حساب (.Account no) مستقل لكل سجل وفى العادة يتم

فهرسة الملف بناء على هذا الحقل المنفرد (Unique). ولكن عند إدخال البيانات فقد يقوم المستخدم بإدخال بيانات نفس السجل مرتين لذلك يستخدم الأمر(SET UNIQUE ON). فعندما يستخدم هذا الأمر قبل فهرسة الملف (Indexing) فإن برنامج (+ DBase III) يقوم بإدخال السجلات المنفردة (Unique) فقط فى الفهرس ويستبعد أى سجلات مكررة. فعند البحث عن سجلات لعرضها أو تعديلها فإن المستخدم لن يرى أى سجلات مكررة. وإذا كانت هناك سجلات مكررة فعلا فإن الفهرس لن يظهرها لأنها ليست جزءا من الملف الفهرسى. كما يمكن تنفيذ هذه العملية عند إنشاء ملف الفهرس كالآتى :

```
INDEX ON name TO Name Unique
```

حيث يؤدى هذا الأمر إلى إنشاء فهرس منفرد (Unique).

ويجب ملاحظة إعادة الوضع المبدئى (Default) عن طريق كتابة الأمر (SET UNIQUE OFF) وذلك عندما يراد استخدام كل السجلات سواء كانت مكررة أو غير مكررة.

ملاحظة

ما سبق ذكره فى هذا الفصل ينطبق أيضا على كل برامج عائلة (DBase) مثل (DBase IV) ، (FoxBase) ، (FoxBase +) ، (FoxPro).

الفصل الثالث والثلاثون

التعامل مع البيانات



عند تعديل بيانات سجل معين أو إضافة سجل جديد فإن من المهم إعطاء المستخدم الفرصة لمراجعة البيانات التى أدخلها وذلك بعرضها على الشاشة ثم إعطائه الإختيار بين تخزين البيانات التى كتبها إذا كانت صحيحة أو تصحيحها مرة ثانية إذا وجد بعض الأخطاء ويتم تنفيذ ذلك باستخدام متغيرات الذاكرة (Memory Variables). وتستخدم هذه المتغيرات كحلقة إتصال بين المستخدم وملف قاعدة البيانات حيث يتم إدخال البيانات التى يكتبها المستخدم أولا فى متغيرات الذاكرة ويقوم البرنامج بتحديد السجل المطلوب تعديله ثم استبدال محتويات كل حقل من حقول هذا السجل بمحتويات متغير الذاكرة المقابل له.

ويمكن توضيح هذه العملية فى خطوات محددة كالآتى :

١ - يتم أولا إنشاء متغيرات الذاكرة (Memory Variables) لاستقبال مدخلات المستخدم وكل متغير من هذه المتغيرات يقابل حقلا معينا من حقول قاعدة البيانات. وفى معظم البرامج يتم إعطاء هذه المتغيرات نفس إسم الحقل مع إضافة الحرف (m) مثل (mname) أو (m_name).

٢ - يتم فتح ملف قاعدة البيانات وملفات الفهرس المرتبطة به ثم يتم استخدام أى أمر من أوامر البحث للوصول إلى السجل المطلوب تعديله مثل (GOTO) ، (LOCATE) ، (FIND) ، (SEEK).

٣ - يتم عرض شاشة الإدخال التى سبق تصميمها ويلاحظ ظهور محتويات الحقول الخاصة بهذا السجل فيقوم المستخدم بتعديل بيانات هذه الحقول.

٤ - فى حالة إضافة سجل جديد يتم عرض شاشة خالية باستخدام الأمر (APPEND BLANK).

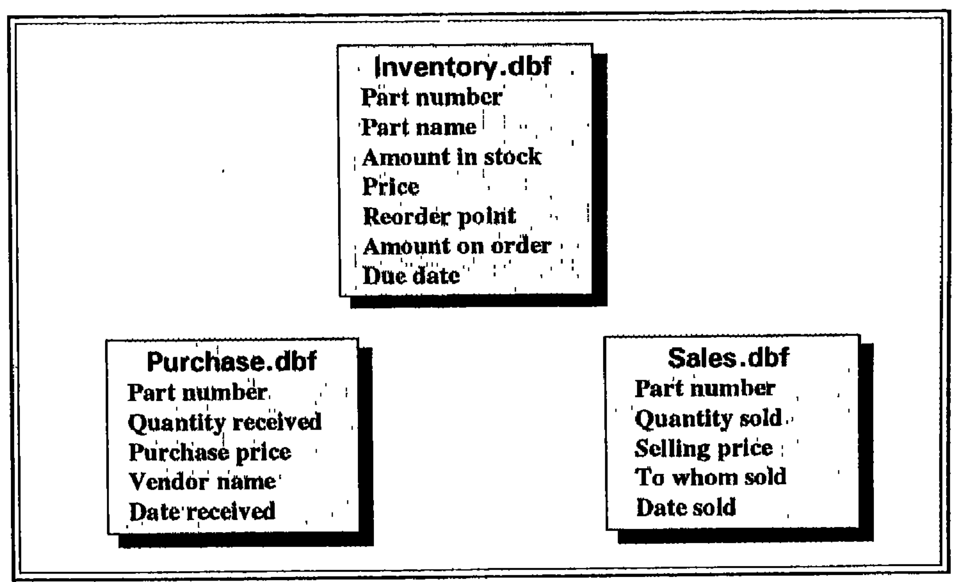
٥ - بعد أن يضيف المستخدم البيانات المطلوبة يتم عرض رسالة (Prompt) لسؤال المستخدم عن صحة البيانات التى أدخلها. واذا لم تكن صحيحة يتم إعطاؤه الفرصة لتعديل هذه البيانات وتصحيح الأخطاء.

٦ - يتم استبدال البيانات الموجودة فى الحقول بالبيانات الموجودة فى متغيرات الذاكرة كل حسب الحقل المقابل له وذلك باستخدام الأمر (REPLACE).

٧ - يتم إغلاق الملف ومسح متغيرات الذاكرة حتى يمكن استخدامها مرة ثانية.

٣٣ - ١ التعديـــل المجمـــع (Batch Updating)

فى كثير من نظم المعلومات يستخدم النظام الطريقة التجميعية فى تعديل البيانات (Batch Updating). ويظهر ذلك بوضوح فى معظم نظم الحسابات (Accounts) والبنوك والمخازن حيث يتم تجميع التعديلات فى ملف منفصل يسمى ملف الحركة (Transaction File) ثم يتم إدخال هذه التعديلات على الملف الرئيسى (Master File) كل فترة. وقد تكون هذه الفترة يوما أو أسبوعا أو شهرا على حسب حجم التعديلات التى يتم إدخالها. ولنأخذ مثلا نظام المخازن (Inventory) كمثال لهذه النظم حيث يتكون نظام المخازن البسيط من ملف رئيسى (Master File) وملفين للحركة (Transaction Files) انظر الشكل (٣٣ - ١)



شكل (٣٣ - ١) نظام المخازن

ويلاحظ من الشكل تركيب الملف الرئيسى (Inventory.dbf) وكذلك تركيب ملفات الحركة (Sales.dbf) و (Purchase.dbf). والملف الرئيسى هنا يحتوى على البيانات الحالية للبضائع المخزنة (Stock) وهو يستمد معلوماته من بيانات حركة البيع (Sales Transactions) التى تصل إليه من ملف المبيعات (Sales.dbf) وكذلك من بيانات حركة الشراء (Purchase transactions) التى تصل إليه من ملف المشتريات (Purchase.dbf). ويلاحظ من الشكل أيضا وجود حقل مشترك بين الملفات الثلاثة وهو حقل رقم الجزء (Part number) وهو الحقل الذى يتم عن طريقه ربط الملفات الثلاثة.

ولكى يتم تعديل الملف الرئيسى من ملف المبيعات مثلا يتم أولا فهرسة الملفات باستخدام حقل رقم الجزء كحقل فهرسى (Index Key) وذلك كالآتى :

```
USE Inventory
INDEX ON part_no TO Master
USE Sales
INDEX ON part_no TO Sales
```

ثم يتم فتح كل ملف فى منطقة عمل مختلفة باستخدام الأمر (SELECT) كما سبق الإيضاح. ويتم استخدام الأمر (UPDATE) فى تعديل حقل الكمية الموجودة مثلا (on_hand) وذلك كالآتى :

```
CLEAR ALL
SELECT 2
USE Sales INDEX Sales
SELECT 1
USE Inventory INDEX Master
UPDATE ON part_no FROM Sales REPLACE ;
on_hand WITH on_hand - B -> Qty
```

ملاحظة

عند زيادة طول السطر عن عرض الشاشة أثناء كتابة البرنامج يتم إضافة الحرف (;) فى نهاية السطر. وهذا الحرف يخبر البرنامج أن المكتوب فى السطر التالى تكملة لهذا السطر ويلاحظ ذلك فى السطر السادس من البرنامج.

ويلاحظ من البرنامج فتح كل ملف فى منطقة عمل (Work Area) مختلفة. كما يلاحظ أن آخر منطقة عمل يتم اختيارها هى التى يتم العمل فيها. وهذا يعنى أنه عند العمل فى أى منطقة يجب أولا إختيار هذه المنطقة باستخدام الأمر (SELECT).

ويلاحظ أيضا استخدام الأمر (UPDATE) فى تعديل الكمية الموجودة فى الحقل (on_hand) فى الملف الرئيسى (Inventory.dbf) عن طريق طرح الكمية الموجودة فى

الحقل (Qty) الخاص بملف المبيعات (Sales.dbf) من الكمية الموجودة فى الحقل (on_hand) فى الملف الرئيسى.

ويجب ملاحظة أن الحرف (B) هنا يمثل منطقة العمل (2) التى تحتوى على ملف المبيعات. كما أن العلامة (<-) تعنى الحصول على محتويات الحقل (Qty) من ملف المبيعات الموجود فى منطقة العمل (B).

ويمكن استخدام نفس الطريقة فى تعديل الملف الرئيسى بناء على بيانات ملف المشتريات (Purchase.dbf) وذلك بكتابة الأمر التالى :

```
UPDATE ON part_no FROM Purchase REPLACE on_hand ;
    WITH  on_hand + B -> Qty
```

ويلاحظ هنا إضافة الكمية التى تم شراؤها إلى الكمية الفعلية الموجودة فى المخزن.

ملاحظة

يتم فى كتاب التطبيقات (الكتاب رقم (٧) فى مجموعة كتب " دلتا ") شرح برنامج مخازن كامل.

٣٣ - ٢ مسح السجلات

يتم مسح السجلات المطلوب مسحها باستخدام الأمر (DELETE) وتتبع فى ذلك نفس الخطوات التى سبق إيضاحها عند تعديل السجلات. حيث يتم أولا البحث عن السجل المطلوب باستخدام أوامر البحث السابق شرحها. أى يتم وضع المؤشر على السجل المطلوب مسحه ثم كتابة الأمر (DELETE) أو يتم تنفيذ ذلك بكتابة سطر واحد كالآتى :

```
DELETE  ALL  FOR  YEAR ( Date )  = myear
```

حيث (myear) هو متغير ذاكرة يتم فيه تخزين الرقم الممثل لسنة معينة. كما أن القيمة (YEAR(Date)) هى القيمة العددية الممثلة للسنة المخزنة فى الحقل (Date) وكلمة (ALL) تحدد المدى (Scope) المطلوب البحث خلاله.

ويؤدى هذا إلى مسح جميع السجلات التى تختص بسنة معينة يقوم المستخدم بتحديدها ويفيد ذلك عندما يراد مثلا مسح بيانات سنين سابقة بعد انتهاء الحاجة اليها. ويجب ملاحظة أن الأمر (DELETE) وحده لايؤدى إلى مسح السجلات مباشرة ولكنه يميز هذه السجلات تمهيدا لمسحها باستخدام الأمر (PACK) فى أى وقت.

ملاحظة

بعض مخططى البرامج يفضلون عدم استخدام الأمر (PACK) يوميا ويقومون بتجميع السجلات المطلوب مسحها كل فترة ثم مسحها مرة واحدة عند التأكد من عدم الحاجة إليها.

وعندما يراد مسح بيانات الملف بالكامل يستخدم الأمر (ZAP) ويجب الحرص عند استخدام هذا الأمر لأنه يمسح السجلات مباشرة دون الحاجة إلى استخدام الأمر (PACK). فمثلا عندما يراد مسح البيانات التى سبق إدخالها فى ملفات المخازن والبدء فى تسجيل بيانات جديدة تستخدم الأوامر التالية :

```
USE Inventory
ZAP
USE Sales
ZAP
USE Purchase
ZAP
CLOSE DATABASES
```

ملاحظة

مسح بيانات الملف لايعنى مسح الملف بالكامل لأن هيكل الملف (File Structure) يظل كما هو بعد المسح.

٣٣ - ٣ نسخ السجلات

قبل مسح بعض السجلات التى انتهت الحاجة اليها قد يكون من المفيد نسخ هذه السجلات فى ملف أرشيف (Archive File). فمثلا فى برنامج المخازن قد تكون البيانات

التى مضى عليها أكثر من خمس سنوات غير مطلوبة ويراد مسحها ولكن يراد الإحتفاظ بها فى ملف أرشيف حتى يمكن الرجوع إليها وقت الحاجة. ويستخدم الأمر (COPY) فى نسخ سجلات من ملف إلى ملف آخر. فمثلا فى برنامج المخازن السابق يمكن كتابة الأوامر التالية :

```
USE Inventory
COPY TO file1  FOR  YEAR ( Date )  =  myear
DELETE  ALL  FOR  YEAR ( Date )  =  myear
PACK
```

وتؤدى هذه الأوامر إلى نسخ جميع السجلات التى تختص بسنة معينة إلى ملف الأرشيف (File1) ثم يتم بعد ذلك مسح هذه السجلات نفسها من ملف قاعدة البيانات الأصلى.

٣٣ - ٤ التعامل مع الملفات المرتبطة

كما سبق الإيضاح فإن الملفات المرتبطة (Related) هى ملفات يربط بينها حقل مشترك. وهى تساعد على تقسيم ملف قاعدة البيانات الكبير إلى عدة ملفات صغيرة نسبيا لتقليل التحميل (Overload) على الذاكرة المؤقتة (RAM) وبالتالى زيادة سرعة التشغيل.

وعندما يراد الحصول على بيانات من عدة ملفات مرتبطة يجب أولا ربط هذه الملفات ببعضها وذلك لأن المؤشر الخاص بكل ملف يكون موضوعا على سجلات مختلفة وبالتالى لايمكن عرض بيانات سجل معين بتجميعها من عدة ملفات إلا بعد وضع المؤشر فى جميع هذه الملفات عند نفس السجل. ولتوضيح ذلك نفرض أن هناك ملفين إسمهما (First.dbf) و (Second.dbf) ونفرض أن الملف (Second.dbf) يحتوى على الحقول (name, birth_d) والملف (First.dbf) يحتوى على الحقل (address) ويراد عرض بيانات هذه الحقول فيتم كتابة الأوامر التالية :

```
SELECT 1
USE First INDEX First
SELECT 2
USE Second INDEX Second
DISPLAY name, birth_d, A -> address
```

حيث (A) هو الحرف المرادف (ALIAS) للمنطقة الأولى (١). فى هذه الحالة يتم عرض بيانات حقول الإسم (name) وتاريخ الميلاد (birth_d) من الملف (Second.dbf) كما يتم عرض العنوان (address) من الملف (First.dbf).

ولكن هل هذه البيانات تختص بنفس السجل ؟ والإجابة على هذا السؤال بالنفى وذلك لأنه لم يتم ربط المؤشر الخاص بالملف (First.dbf) بالمؤشر الخاص بالملف (Second.dbf) وبالتالى ليس هناك ما يضمن وقوف المؤشر على نفس السجل فى الملفين. ولكى يتم ربط الملفات ببعضها يستخدم الأمر (SET RELATION).

٣٣ - ٥ إستخدام الأمر (SET RELATION)

هذا الأمر يربط بين ملفين عن طريق حقل مشترك فيهما ويؤدى هذا إلى ربط المؤشر الخاص بالملف الثانى بالمؤشر الخاص بالملف الاول. وبعد تحريك المؤشر فى الملف الأول للوصول إلى سجل معين باستخدام أوامر البحث السابق شرحها فإن المؤشر الخاص بالملف الثانى يتحرك معه تماما وبالتالى فعند إجراء أى عملية على بيانات هذا السجل يمكن الحصول على بياناته من الملفين فى نفس الوقت.

فمثلا فى الملفين (First.dbf) و (Second.dbf) نفرض أنه يوجد حقل مشترك وهو حقل (emp_no) أى رقم الموظف فإن الربط بين الملفين يتم عن طريق كتابة الأوامر التالية :

```
SELECT 1
USE First INDEX First
SELECT 2
USE Second INDEX Second
SET RELATION TO emp_no INTO First
DISPLAY name, birth_d, A -> address
```

ويجب ملاحظة أنه لايمكن استخدام أكثر من علاقة واحدة فى كل منطقة عمل (Work area). كما أن هذه العلاقة تنتهى بمجرد إغلاق منطقة العمل أو استخدام الأمر (SET RELATION TO) دون كتابة أى شىء بعده. كما يجب ملاحظة أن الملف الذى يتم ربطه والذى يكتب إسمه بعد (INTO) يجب أن يكون مفهرسا (Indexed) على الحقل المشترك بين الملفين أما الملف الآخر فلا يشترط أن يكون مفهرسا.

1	2	3
First.dbf Name Birth-d Emp-no	Second.dbf Address Emp-n	

شكل (٣٣ - ٢)

وإذا رجعنا إلى البرنامج السابق فإننا نلاحظ أن منطقة العمل (2) هى منطقة العمل الفعالة (Active) التى يجرى العمل عليها وذلك لأنها آخر منطقة عمل تم اختيارها بواسطة الأمر (SELECT). لذلك فإن أى حركة لمؤشر الملف فى هذه المنطقة يتبعها نفس الحركة لمؤشر الملف فى المنطقة الأولى وبالتالى يمكن الحصول على بيانات لنفس السجل من الملفين.

ملاحظة

فى حالة إنشاء علاقة بين ملفين يجب أن يكون الحقل المشترك منفردا (Unique) أى غير متكرر.

٣٣ - ٦ استخدام ملف المنظر (View File)

يستخدم ملف المنظر (View File) فى ربط الملفات كما سبق الإيضاح فى الجزء الخاص ببرنامج المساعد (Assistant). وهو يساعد على تخزين العلاقة بين ملفين فى ملف منفصل. وهذا عكس استخدام الأمر (SET RELATION TO) الذى يؤدى إلى ربط الملفين ربطا مؤقتا. ويمكن إنشاء عدة ملفات منظر كل منها يحتوى على مجموعة من الحقول المطلوب عرضها على الشاشة أو طباعتها على الطابعة. وبمعنى آخر يمكن النظر إلى قاعدة البيانات من زوايا مختلفة (Views).

ويمكن إنشاء ملفات المنظر من برنامج المساعد (Assistant) كما سبق الإيضاح أو بكتابة الأمر (CREATE VIEW) كما يمكن تعديلها أو إنشاؤها بواسطة الأمر

(MODIFY VIEW) وفى هذه الحالة تظهر نفس الشاشات التى سبق إيضاحها فى استخدام برنامج المساعد (Assistant).

كما يمكن إنشاء ملف المنظر من خلال البرنامج عن طريق كتابة الأمر (CREATE VIEW FROM INVIRONMENT) وذلك بعد إدخال العلاقة المطلوبة عن طريق الأمر (SET RELATION TO). وفى جميع الأحوال يتم استخدام هذا الملف عن طريق كتابة الأمر (SET VIEW TO) ثم كتابة إسم الملف. وفى المثال السابق مثلا يمكن كتابة الآتى :

```
SET RELATION TO emp_no INTO first
CREATE VIEW F_second FROM INVIRONMENT
```

ويؤدى هذا إلى إنشاء ملف المنظر (F_second). هذا الملف يحتوى على الآتى :

١ - كل ملفات قواعد البيانات وملفات الفهرس ورقم كل منطقة عمل (Work Area).
٢ - العلاقة أو العلاقات الموجودة بين الملفات.
٣ - ملف التشكيل (Format File) المفتوح.

ويمكن التحكم فى أسماء الحقول التى تظهر فى هذا الملف باستخدام الأمر (SET FIELDS TO) ثم كتابة أسماء الحقول المطلوب عرض بياناتها. وهذا الأمر يؤدى إلى تصفية حقول الملفات الموجودة فى ملف المنظر بحيث لاتظهر إلا بيانات الحقول التى يتم تحديدها.

ويجب ملاحظة استخدام الأمر (SET FIELDS ON) عندما يراد استخدام هذه الحقول التى تم اختيارها. ويمكن إضافة ذلك إلى المثال السابق فيصبح كالآتى :

```
SET RELATION TO emp_no INTO first
SET FIELDS TO name, birth_d, A -> address
CREATE VIEW F_second FROM INVIRONMENT
SET FIELDS ON
DO WHILE .NOT. EOF()
   DISPLAY
   SKIP
ENDDO
```

ويمكن إلغاء عملية تصفية الحقول باستخدام الأمر (SET FIELDS OFF) كما يمكن تنفيذ ذلك أيضا باستخدام الأمر(CLEAR FIELDS) أو(SET FIELDS TO) دون كتابة أي شيء بعده.

مـلاحظة

ما سبق ذكره في هذا الفصل ينطبق أيضا على كل برامج عائلة (DBase) مثل (DBase IV) ، (FoxBase +) ، (FoxPro).

الفصل الرابع والثلاثون

الطباعة



هناك أوامر تساعد على توجيه مخرجات البرنامج إلى الشاشة أو الطابعة حسب الحاجة ولكن فى جميع الأحوال يجب التأكد أولا أن الطابعة قد تم تشغيلها وتوصيلها بالجهاز.

٣٤ - ١ أوامـــر الطبـاعـة

هناك أمران يستخدمان فى توجيه البيانات إلى الطابعة. ورغم أنهما يحققان نفس الهدف إلا أنهما يختلفان فى خصائصهما ولذلك يجب الإلمام بخصائص كل منهما حتى يمكن معرفة أى الأمرين يجب استخدامه وفى أى مكان من البرنامج. وهذان الأمران هما (SET DEVICE TO PRINT) و (SET PRINT ON).

٣٤ - ٢ استخدام الأمر (SET DEVICE TO PRINT)

يستخدم هذا الأمر بصفة خاصة عندما تكون هناك أوامر (@..SAY) يتم من خلالها عرض البيانات. فعند استخدام الأمر (SET DEVICE TO PRINT) يتم توجيه هذه البيانات إلى الطابعة بدلا من الشاشة. وهذا يساعد على التحكم فى شكل ومكان البيانات على الورقة المطبوعة حيث يمكن عن طريق كتابة الإحداثيات بعد (@) تحديد مكان الطباعة على الورقة كما يتم بالنسبة للشاشة تماما ويسمى ذلك الطباعة المشكلة (Formatted). ويجب ملاحظة أن هذا الأمر يؤدى إلى توجيه البيانات إلى الطابعة مع عدم ظهورها على الشاشة كما يجب التنبيه إلى استخدام الأمر (SET DEVICE TO SCREEN) بعد انتهاء الطباعة.

٣٤ - ٣ استخدام الأمر (SET PRINT ON)

يستخدم هذا الأمر مع الطباعة غير المشكلة (Unformatted) أى البيانات التى يتم عرضها من خلال الأوامر (?? ,? ,DISPLAY, LIST) ولكن لايستخدم مع الأمر (@...SAY). وهو يؤدى إلى طباعة البيانات على الطابعة مع ظهورها على الشاشة فى نفس الوقت إلا فى حالة استخدام الأمر (SET CONSOLE OFF) الذى يؤدى إلى عدم ظهورها على الشاشة.

والأمر (SET PRINT ON) يسمح بإضافة بعض الإمكانيات الخاصة بالطباعة مثل طباعة الحروف المائلة (Italic) أو المضغوطة (Condensed).

٣٤ - ٤ التحويل بين الشاشة والطابعة

كما سبق الإيضاح فإن استخدام الأمر (SET DEVICE TO PRINT) يؤدى إلى توجيه السطور (@..SAY) إلى الطابعة بدلا من الشاشة وبالتالى لاتظهر هذه السطور على الشاشة أثناء الطباعة. ولذلك يمكن لمخطط البرامج استغلال الشاشة أثناء الطباعة فى عرض بعض الرسائل للمستخدم مثل إضافة ورقة جديدة للطابعة مثلا وذلك عن طريق التحويل بين الشاشة والطابعة حسب الحاجة. ولتوضيح ذلك يمكن ملاحظة الأوامر التالية :

```
CLEAR
@ 10,10 SAY "Insert paper correctly"
?
WAIT SPACE(10) + "Press any key to begin printing"
SET DEVICE TO PRINT
---------
---------
---------      printing
---------
---------
SET DEVICE TO SCREEN
CLEAR
? CHR(7)
@ 10,10 SAY "Insert another piece of paper"
?
WAIT SPACE(10) + "and press any key to begin again"
SET DEVICE TO PRINT
```

ويلاحظ هنا أن البرنامج ينبه المستخدم إلى وضع ورقة جديدة فى الطابعة بعد انتهاء صفحة الطباعة. وتم استخدام الجرس لتنبيه المستخدم عند انتهاء الصفحة عن طريق الدالة (CHR(7)).

ولكن كيف يستطيع البرنامج تحديد نهاية صفحة الطباعة ؟.

يستطيع البرنامج تنفيذ ذلك عن طريق الإحداثيات التى يتم كتابتها فى الأمر (SAY...@). فعند كتابة رقم سطر أصغر من رقم سطر سابق تقوم الطابعة بالبداية من صفحة جديدة عند هذا السطر. فمثلا عند كتابة السطرين التاليين :

```
@ 10,10 SAY "-------"
@ 9,10  SAY "-------"
```

فإن ذلك يؤدى إلى القفز (Eject) إلى ورقة جديدة ثم التحرك إلى السطر رقم (٩) لكتابة السطر التالى. لذلك يمكن عن طريق البرنامج التحكم فى عدد السطور التى تظهر فى الورقة وتوقيت الإنتقال إلى الصفحة التالية. ويمكن أيضا القفز إلى الصفحة التالية باستخدام الأمر (EJECT).

تحـذير

الأمر (EJECT) يعمل بصورة جيدة عندما يكون قد سبق الطباعة من أول الورقة تماما.

٣٤ - ٥ تحديد الهامش الأيسر

يتم تحديد الهامش الأيسر أولا بتحديد مكان العمود رقم صفر كما يظهر على الورقة. ويمكن تنفيذ ذلك عن طريق كتابة البرنامج الصغير التالى :

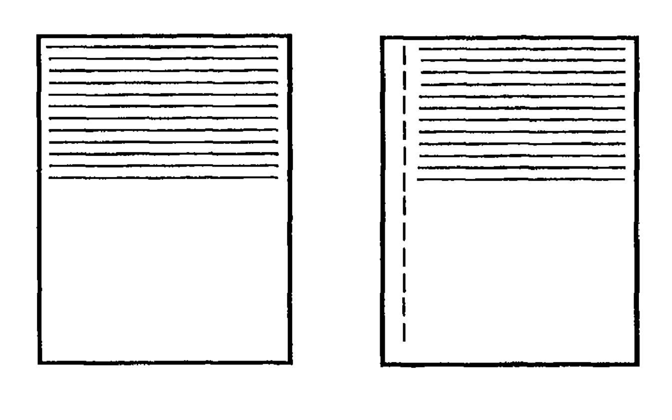
```
SET DEVICE TO PRINT
@ 0,0 SAY 'Testing...'
EJECT
SET DEVICE TO SCREEN
```

ثم وضع الورقة على الطابعة وتنفيذ هذا البرنامج ومن خلال ملاحظة بدء كتابة كلمة (Testing) على الطابعة يمكن تحديد مكان الكتابة المقابل للعمود صفر على شاشة الحاسب.

كما يمكن استخدام الأمر (SET MARGIN TO) فى تغيير مكان الكتابة بالنسبة للمكان السابق تحديده. أي يمكن بواسطة هذا الأمر التحكم فى المسافة الكلية بين بداية الكتابة وبداية الورقة من اليسار. فمثلا عند كتابة الأمر

SET MARGIN TO 20

فى هذه الحالة يلاحظ التغيير فى شكل ورقة الطباعة كما يتضح من الشكل.



شكل (٣٤ - ١)

ولتحديد الهامش العلوى (Top Margin) يمكن استخدام نفس البرنامج السابق حيث يمكن ملاحظة أول سطر فى الطباعة والذى يقابل السطر صفر على الشاشة. ولتحديد هذا الهامش يمكن إضافة عدة سطور خالية قبل بداية التقرير المكتوب على الشاشة وكذلك يمكن ترك عدة سطور خالية أسفل التقرير حتى يتم التحكم فى الهامش السفلى (Bottom Margin).

٣٤ - ٦ طباعة السطر الأخير من التقرير

تحتوى الطابعة على مخزن ذاكرة مؤقت (Buffer) يقوم بتخزين سطر الطباعة تخزينا مؤقتا قبل انتقاله إلى ورقة الطباعة. ولايتم انتقال هذا السطر من المخزن المؤقت (Buffer) إلا عند إدخال شفرة الإدخال (Carriage Return Code) وهذه الشفرة هى كود الآسكى (13). وفى الوضع العادى يتم إدخال هذه الشفرة بعد نهاية السطر على الشاشة عند الضغط على مفتاح الإدخال. ولكن برنامج (+DBase III) يرسل هذه الشفرة

فى بداية السطر لذلك فإن طباعة جميع سطور التقرير لايحدث فيها أى مشكلة أما السطر الأخير فهناك احتمال عدم طباعته على ورقة الطباعة. ويمكن التغلب على هذه المشكلة عن طريق استخدام الأمر (EJECT) حيث أن هذا الأمر يرسل شفرة الإدخال (Carriage Return Code) بالإضافة إلى نقل الصفحة.

كما يمكن التغلب على هذه المشكلة أيضا عن طريق كتابة السطور التالية فى نهاية التقرير :

```
SET PRINT ON
?? CHR(13)
SET PRINT OFF
```

أو كتابة السطور التالية :

```
SET DEVICE TO PRINT
@ 23,0 SAY CHR(13) && Sends carriage return code
SET DEVICE TO SCREEN
```

ملاحظة

هذه المشكلة تظهر فقط عند طباعة التقرير من خلال البرنامج ولكنها لاتظهر عند طباعة التقرير من خلال برنامج المساعد (Assistant) أو من خلال الأمر (CREATE / MODIFY REPORT).

٣٤ - ٧ إدخال بعض المؤثرات الخاصة (Special Effects)

يمكن عن طريق إدخال شفرة تحكم (Control Code) إلى الطابعة الحصول على مؤثرات خاصة تزيد من كفاءة الطباعة. وتبدأ شفرة التحكم (Control Code) فى معظم الطابعات بشفرة مفتاح الهروب (Escape Key) وهى كود الآسكى (ASCII 27) ويليها أى شفرة أخرى ولذلك تسمى شفرة التحكم عادة شفرة الهروب (Escape Code). فمثلا لكى تستخدم الطابعة فى طباعة حروف مائلة (Italic) يجب إدخال شفرة التحكم (ESC F). وتتم كتابة ذلك فى البرنامج كالآتى :

```
SET PRINT ON
?? CHR(27) + "F"
```

كما يمكن إنهاء كتابة الحروف المائلة عن طريق كتابة الأوامر التالية فى البرنامج :

```
SET PRINT ON
?? CHR(27) + "G"
```

ملاحـظة

يجب التأكد من إعادة الطابعة إلى الوضع المبدئى (Default) بعد انتهاء إستخدام المؤثر الخاص الذى تم إدخاله. وإذا لم يحدث ذلك فإن الطابعة ستظل تعمل بهذا التأثير الخاص.

٣٤ - ٨ تحديد مكان انتقال الصفحة (Page Break)

عندما تكون التقارير المطلوب طباعتها طويلة فإن من المهم التحكم فى طول صفحة الطباعة بحيث لاتزيد مثلا عن ٦٠ سطرا. كما يتم الإنتقال إلى الصفحة التالية آليا. ويمكن تنفيذ ذلك من خلال البرنامج التالى :

```
SET TALK OFF
STORE 61 TO tline
STORE 5 TO tcolumn
STORE 0 TO pagenum
USE Cadets INDEX name
GO TOP
SET DEVICE TO PRINT
DO WHILE .NOT. EOF()
  IF tline > 60
    STORE 1 TO tline
    STORE pagenum + 1 TO pagenum
    @ tline,tcolumn + 66 SAY "page" + STR(pagenum,3)
    @ tline + 1 , tcolumn + 66 SAY DATE()
```

```
        @tline + 4 ,tcolumn + 30 SAY 'Names and phone ;
                numbers'
    ENDIF
    @ tline, tcolumn SAY TRIM(name) + '' + Phone
    SKIP
    STORE tline + 1 to tline
ENDDO
EJECT
SET DEVICE TO SCREEN
RETURN
```

ويلاحظ من هذا البرنامج أنه تم إنشاء متغير الذاكرة (tline) ليمثل رقم السطر فى الصفحة وتم إدخال القيمة (61) فيه. كما تم إنشاء متغير الذاكرة (tcolumn) ليمثل رقم العمود الذى يمثل وضع الحرف على السطر عند كتابة الأمر (@...SAY) وتم إدخال القيمة (5) فى هذا المتغير. كما تم إنشاء متغير الذاكرة (Pagenum) ليمثل رقم الصفحة وتم إدخال القيمة صفر فى هذا المتغير. ثم تم استخدام الأمر (IF) مع وضع الشرط (Tline>60) بعده. فعندما يزيد عدد السطور عن ٦٠ سطرا يتم تنفيذ الأوامر التالية وهذا يؤدى إلى البدء من صفحة جديدة كما يؤدى إلى زيادة رقم الصفحة واحدا. وإذا لم يصل عدد السطور إلى ٦٠ سطرا يتم تنفيذ الأوامر بعد (ENDIF) والتى تؤدى إلى كتابة البيانات المطلوبة ثم الإنتقال إلى سجل جديد باستخدام الأمر (SKIP) مع زيادة رقم السطر واحدا. ويلاحظ فى نهاية البرنامج استخدام الأمر (EJECT) للتأكد من طباعة السطر الأخير ثم إعادة الطابعة إلى الوضع المبدئى (Default).

ملاحظة

ما سبق ذكره فى هذا الفصل ينطبق أيضا على كل برامج عائلة (DBase) مثل (DBase IV) ، (FoxBase +) ، (FoxPro).



الفصل الخامس والثلاثون

التعامل مع بيئة الحاسب



تطلق كلمة بيئة الحاسب (Environment) للدلالة على المكونات الخارجية التى تتعامل معها وحدة التشغيل المركزية (Central Processing Unit). وتقاس كفاءة البرنامج المكتوب للحاسب بقدرته على التعامل مع هذه المكونات والتحكم فيها بما يحقق أحسن استغلال لخصائصها الفنية. ويتضمن هذا الفصل شرح الوسائل المختلفة للتحكم فى بيئة الحاسب (Environment) من خلال البرنامج. كما يتضمن إستخدام بعض الدوال المتقدمة (Advanced Functions) لتحقيق أحسن إستغلال لهذه المكونات.

٣٥ - ١ التعامل مع القرص

فى العادة يتم فتح ملف قاعدة البيانات (DBase File) من خلال البرنامج ولكن فى بعض الأحيان قد يتطلب الأمر أن يدخل المستخدم إسم الملف المطلوب إستخدامه خصوصا فى البرامج التى تتصف بالكفاءة والقدرة على التعامل مع ملفات مختلفة وحقول بيانات مختلفة. والبرنامج يتيح للمستخدم إستخدام الدالة (FILE()) التى يستطيع المستخدم من خلالها إدخال إسم الملف المطلوب.

وهذه الدالة تعطى القيمة صحيح (True) أو غير صحيح (False) حسب وجود الملف المكتوب بين القوسين فى القرص أو عدم وجوده مع ملاحظة أن إسم الملف يكون متضمنا الإمتداد (Extension). ويمكن من خلال البرنامج إضافة هذا الإمتداد(DBF.) حتى يستطيع المستخدم إدخال إسم الملف دون أى ارتباك نتيجة عدم معرفة إمتداد الملف.

والبرنامج التالى يوضح استخدام هذه الدالة :

```
mfile = SPACE(8)
CLEAR
@ SAY 'Enter the file name:' GET mfile PICTURE '@!' READ
mfile = LTRIM(TRIM(mfile)) + '.DBF'
IF FILE(mfile)
        -----
        -----  comands
        -----
ENDIF
```

وهذا البرنامج يقوم بإنشاء متغير ذاكرة (mfile) طوله ٨ حروف لإدخال إسم الملف فيه ثم يقوم بعرض رسالة للمستخدم ليدخل إسم الملف المطلوب ثم يقوم بالتخلص من المسافات الخالية (Spaces) ويضيف الإمتداد (DBF) إلى الإسم. ويستخدم الأمر (IF) لاختبار وجود هذا الملف على القرص فإذا وجده ينفذ الأوامر التالية.

ويجب ملاحظة أن هذا البرنامج يقوم بالبحث خلال وحدة الأقراص الحالية (Current Drive) فإذا أريد إدخال مسار معين (Path) يتم إدخال هذا المسار أولا فى متغير ذاكرة وذلك كالآتى مثلا :

```
STORE ' C:\ DBase\ Cadets\ ' TO Path
IF FILE(path + mfile)
        ------
        ------
        ------ commands
        ------
        ------
INDIF
```

كما يمكن استخدام الدالة (DBF()) فى معرفة إسم ملف قاعدة البيانات المفتوح فى منطقة العمل الحالية (Current Work Area) وكذلك تستخدم الدالة (NDX()) فى معرفة أسماء ملفات الفهرس المستخدمة فى منطقة العمل الحالية.

وحيث أن برنامج (DBase III+) يسمح بفتح حتى ٧ ملفات فهرس مع كل ملف قاعدة بيانات لذلك فإن الدالة (NDX()) تعطى إسم الملف حسب الرقم الذى يتم إدخاله بين القوسين. فمثلا الأمر التالى يعطى إسم ملف الفهرس الرئيسى :

```
? NDX(1)
```

وتستخدم الدالة (FIELD()) فى تحديد إسم أى حقل داخل ملف قاعدة البيانات من خلال البرنامج. وهذه الدالة تقبل أى رقم من ١ إلى ١٢٨ بين القوسين وهذا الرقم يمثل ترتيب الحقل المطلوب داخل الملف. فمثلا الأمر التالى يوضح إسم أول حقل فى الملف :

```
? FIELD(1)
```

وهذه الدوال تساعد مخطط البرامج على كتابة برنامج يستطيع التعامل مع ملفات مختلفة بتركيب (Structure) مختلف لكل ملف.

وهناك دالة أخرى تستخدم فى تحديد آخر تاريخ تم تعديل الملف فيه وهى الدالة (LUPDATE()). وهذه الدالة تكون مفيدة للتأكد من عدم التكرار (Duplication) لبعض العمليات مثل التجميع اليومى للكميات (Daily Totaling).

٣٥ - ٢ تحديد حجم الملف وحجم القرص المستخدم

يمكن من خلال البرنامج التحكم فى حجم الملف المستخدم وتنبيه المستخدم عندما يزيد هذا الحجم بدرجة كبيرة حتى يقوم بالتخلص من بعض السجلات التى قد لاتكون هناك حاجة إليها. وتستخدم لذلك الدالة (RECCOUNT()) هذه الدالة تعطى العدد الكلى للسجلات داخل الملف والبرنامج التالى يوضح استخدامها :

```
USE Cadets
IF RECCOUNT() > 2000
    ? CHR(7)
    @ 10,10 SAY 'Please delete some records'
ENDIF
```

كما يمكن أيضا تحديد المساحة التخزينية التى يحتاجها الملف وذلك باستخدام الدالة (RECSIZE()). هذه الدالة تعطى المساحة التخزينية التى يحتاجها السجل الواحد من الملف أى أنها تعطى مجموع المساحات التخزينية للحقول بالحروف (Bytes). وحتى يمكن تحديد المساحة التخزينية للملف يتم ضرب عدد السجلات فى المساحة التخزينية للسجل الواحد كالآتى مثلا :

```
STORE  RECCOUNT * RECSIZE  TO  size
```

وهذا فى الواقع لايعطى المساحة التخزينية الكلية للملف حيث أن هناك مساحة أخرى تستخدم كمعلومات تقديمية (Header Information). هذه المعلومات التقديمية تساعد على تتبع أسماء الحقول وأطوالها وأنواعها. ولذلك يجب إضافة المساحة التخزينية لهذه

المعلومات حتى يتم حساب المساحة التخزينية الكلية بدقة.

وحتى يتم حساب المساحة التخزينية للمعلومات التقديمية (Header Information) يجب أولا معرفة عدد الحقول داخل السجل وذلك عن طريق الأوامر التالية :

```
USE Cadets
numfields = 0
null = ""
DO WHILE null < FIELD(numfields + 1)
    numfields = numfields + 1
ENDDO
USE
```

والحلقة التكرارية فى هذا البرنامج تؤدى إلى زيادة عدد الحقول بواحد طالما كان إسم الحقل أكبر من السلسلة الحرفية الخالية (null). وحيث أن السلسلة الحرفية الخالية قيمتها صفر فإن أى حروف موجودة فى إسم الحقل تؤدى إلى زيادة عدد الحقول بواحد. وهكذا يعطى هذا البرنامج العدد الكلى للحقول فى الملف.

ولحساب المساحة التخزينية للمعلومات التقديمية تستخدم العلاقة التالية :

```
header = (32 * numfields) + 34
```

ومن هذا يمكن حساب المساحة التخزينية الكلية للملف عن طريق العلاقة التالية :

```
totalsize = size + header + 1
```

حيث (Size) هو الحجم الذى سبق تحديده عن طريق ضرب عدد السجلات (RECCOUNT()) فى المساحة التخزينية للسجل(RECSIZE()). كما أن الرقم (١) تم إضافته فى المعادلة ليمثل علامة نهاية الملف (End of file marker) والتى تحتل حرفا واحدا (One byte).

ومعرفة المساحة التخزينية الكلية للملف تكون فى بعض الأحيان عملية مهمة جدا. فمثلا عندما يراد عمل فرز للملف (Sorting) فالمعروف أن عملية الفرز تؤدى إلى إنشاء ملف آخر غير الملف الأصلى. أى أنه يلزم أولا التأكد أن المساحة التخزينية للقرص تزيد عن

ضعف المساحة التخزينية لملف قاعدة البيانات. لذلك يمكن استخدام البرنامج التالى لتنفيذ هذه العملية.

```
USE Cadets
IF DISKSPACE < totalsize * 2
  ? CHR(7)
  @ 10,10 SAY "Not enough room in disk to sort this file"
ELSE
  SORT ON grade DESCENDING TO Temp
ENDIF
```

ويؤدى هذا البرنامج إلى اختبار المساحة التخزينية للقرص عن طريق الدالة (DISKSPACE()) فإذا كانت أقل من ضعف المساحة التخزينية لملف قاعدة البيانات يتم تنبيه المستخدم حتى لايقوم بإجراء عملية الفرز. ويلاحظ هنا إستخدام الدالة(CHR(7)) لتنبيه المستخدم عن طريق تشغيل الجرس (Bell).

ويمكن استخدام الدالة (DISKSPACE()) فى عمل نسخة إحتياطية من ملف قاعدة البيانات الكبير والمخزن على القرص الصلب وذلك بنسخ الملف فى مجموعة من الأقراص المرنة. ويمكن تنفيذ ذلك عن طريق نسخ مجموعة من السجلات إلى كل قرص مرن حسب السعة التخزينية لهذا القرص. ويتم ذلك عن طريق البرنامج التالى :

```
USE Cadets
SET DEFAULT TO B
DO WHILE .NOT. EOF()
     WAIT"insert new disk in drive B,and press a key"
     COPY NEXT(DISKSPACE()-(headersize))/RECSIZE() ;
        TO Backup
     SKIP
ENDDO
USE
```

ويلاحظ من خلال الحلقة التكرارية أن البرنامج يستمر فى نسخ السجلات واحدا تلو الآخر إلى القرص المرن الموجود فى وحدة الأقراص (B) طالما كانت النسبة بين المساحة

التخزينية للقرص ناقصا المساحة التخزينية للمعلومات التقديمية (Header Information) وبين المساحة التخزينية للسجل الواحد أكبر من (١) وعندما تصبح هذه النسبة أقل من واحد (وهذا يعنى أن المساحة التخزينية المتبقية على القرص لاتكفى لتخزين سجل) فإن السجل التالى لايتم نسخه ويتوقف تنفيذ الحلقة التكرارية حتى يقوم المستخدم بوضع قرص جديد لتخزين مجموعة أخرى من السجلات وهكذا.

٣٥ - ٣ مسح وتغيير إسم الملف

يمكن من خلال البرنامج إعطاء المستخدم الفرصة لمسح الملف أو تغيير إسمه حسب الحاجة. ويجب فى هذه الحالة إضافة الإمتداد إلى إسم الملف ويمكن استخدام الطريقة السابق شرحها باستخدام الدالة (FILE()). كما يمكن استخدام البرنامج التالى لمساعدة المستخدم على كتابة الإسم متضمنا الإمتداد.

```
File = SPACE(12)
@ 10,10 SAY 'Enter name of file to delete'
@ 12,10 SAY 'including file extension'
@ 14,5 GET file PICTURE '!!!!!!!! . !!!'
READ
IF FILE(file)
    ERASE & file
ELSE
    ? CHR(7)
@ 18,10 SAY 'There is no file by that name'
```

وهذا البرنامج يجبر المستخدم على إضافة الإمتداد(Extension) فى نهاية إسم الملف. ويلاحظ هنا استخدام الحرف(&) الذى يسمى التعويض بالماكرو (Macro Substitution) والذى يتم من خلاله وضع إسم الملف الذى أدخله المستخدم بعد الأمر (ERASE).

ويمكن استخدام الأمر (RENAME) أيضا لتغيير إسم الملف ويمكن فى هذه الحالة إستخدام نفس البرنامج وذلك عن طريق كتابة الأمر التالى بدلا من الأمر (ERASE).

```
RENAME & file TO Cadets
```

٣٥ - ٤ تعديل تركيب الملف (Modifying Structure)

عندما يراد تعديل تركيب ملف قاعدة البيانات فإن من المهم السيطرة على هذه العملية من خلال البرنامج. وذلك لأن تعديل هذا التركيب قد يؤدى إلى توقف البرنامج عندما يتم بواسطة المستخدم دون توجيه أو إشراف من البرنامج.

ولتنفيذ ذلك يتم أولا نسخ تركيب الملف (Structure of dbase file) فى ملف مؤقت (Temporary) مع استخدام الإختيار (Extended). هذا الإختيار يؤدى إلى تحويل تركيب الملف إلى مجموعة من السجلات أى أن كل حقل يصبح سجلا فى الملف الجديد. وهذا يساعد على مسح أو تعديل أى حقل بنفس الطريقة التى يتم بها تعديل أو مسح أى سجل فى ملف قاعدة البيانات. وعند الإنتهاء من إدخال التعديلات المطلوبة يتم إنشاء ملف جديد بالتركيب المطلوب وذلك عن طريق الأمر (CREATE FROM).

فمثلا لتنفيذ هذه العملية على الملف (Cadets) الذى يحتوى على الحقول التالية :

Field	Field Name	Type	Width	Dec
1	name	Character	30	
2	address	Character	30	
3	age	Numeric	4	1
4	grade	Numeric	5	2

يتم أولا نسخ هذا التركيب (Structure) فى ملف مؤقت (Temporary) يتم تسميته (Temp) مثلا مع استخدام الإختيار (EXTENDED). وفى هذه الحالة يتم تحويل الحقول إلى سجلات ويكون كل سجل منها مكونا من أربعة حقول هى إسم الحقل (Field_name) ، ونوع الحقل (Field_Type) ، وطول الحقل (Field_Len) ، وعدد الأرقام العشرية (Field_Dec). ويصبح حقل الإسم هو السجل الأول فى قاعدة البيانات وحقل العنوان هو السجل الثانى ... وهكذا.

فمثلا لكى يتم مسح حقل الدرجة (Grade) يتم كتابة الأمر التالى فى البرنامج :

```
DELETE RECORD 4
```

كما يمكن تعديل طول أى حقل مثلا باستخدام الأمر التالى :

REPLACE Field_Len WITH 35 FOR Field_name = "address"

وذلك يؤدى إلى تعديل طول حقل العنوان من ٣٠ الى ٣٥ حرفا.

وبعد الإنتهاء من إدخال جميع التعديلات المطلوبة على الحقول من خلال البرنامج يتم استخدام الأمر (PACK) لتخزين هذه التعديلات ثم يتم استخدام الأمر (CREATE FROM) لإعادة الملف المؤقت (Temp) من الصورة الممتدة (EXTENDED) إلى الصورة الطبيعية أى تحويل السجلات إلى حقول مرة ثانية وذلك بكتابة الأمر التالى :

CREATE New FROM Temp

ويؤدى هذا إلى إنشاء ملف جديد إسمه (New) بالتركيب المطلوب. ويجب ملاحظة أن التعديلات التى تم إدخالها تتعلق بتركيب قاعدة البيانات فقط ولكن السجلات مازالت كما هى. ولكى يتم تعديل السجلات على التركيب الجديد يتم استخدام الأمر التالى :

APPEND FROM Cadets

ويجب التأكد قبل ذلك من فتح الملف (New) باستخدام الأمر (USE) ثم يتم إغلاق الملف (New) الذى يتم تغيير إسمه بعد ذلك بواسطة الأمر (RENAME) إلى (Cadets) مرة ثانية.

٣٥ - ٥ خطوات إنهاء البرنامج

قبل إنتهاء البرنامج هناك مجموعة من الأوامر التى يتم كتابتها لإغلاق الملفات المفتوحة ومسح متغيرات الذاكرة ثم العودة إلى الوضع المبدئى للبرنامج وهى تختلف حسب نوع البرنامج وهل هو فرعى أو رئيسى.

ففى حالة البرنامج الفرعى يتم الرجوع إلى البرنامج الذى قام باستدعاء هذا البرنامج الفرعى بواسطة الأمر (RETURN) وهذا الأمر يقوم بمسح جميع متغيرات الذاكرة

الخاصة (Private) ولكنه لايمسح المتغيرات العامة (Public). ولكى يتم مسح المتغيرات العامة يستخدم الأمر (CLEAR MEMORY) ولكن يجب الحذر عند استخدام هذا الأمر لأنه يمسح كل الذاكرة المؤقتة. وقد تكون بعض البرامج الأخرى فى حاجة إلى بعض المتغيرات المخزنة بها.

٣٥ - ٥ - ١ إغلاق الملفات

من المهم جدا قبل إنهاء البرنامج التأكد من إغلاق ملفات قواعد البيانات بصفة خاصة. لأن عدم إغلاقها قد يؤثر فى تكامل قاعدة البيانات (Integrity). ويستخدم الأمر التالى لإغلاق ملفات قواعد البيانات :

```
CLOSE DATABASES
```

وهذا الأمر لايغلق ملفات قواعد البيانات فقط ولكنه يغلق أيضا ملفات الفهرس (Index) والتشكيل (Format) وباقى الملفات المرتبطة بها.

٣٥ - ٥ - ٢ العودة إلى البيئة المبدئية (Default Environment)

من المهم أيضا قبل إنهاء البرنامج التأكد من العودة إلى الأوضاع المبدئية (Default) لبرنامج (+DBase III) حتى لاتؤثر الأوضاع التى تم إدخالها على أى برامج أخرى يتم كتابتها. ويتم ذلك عن طريق تعديل كل وضع للأمر (SET) إلى عكس الحالة التى تم إدخالها. فمثلا عند كتابة الأمر (SET TALK OFF) فى بداية البرنامج يتم كتابة الأمر (SET TALK ON) فى نهاية البرنامج وهكذا.

والسطور التالية توضح الأوامر التى يتم استخدامها عادة فى نهاية معظم البرامج :

```
SET TALK ON
SET ESCAPE ON
SET BELL ON
SET HEADING ON
SET HELP ON
SET SEFETY ON
```

```
SET STATUS ON
CLEAR ALL
CLEAR
RETURN
```

ملاحظة

ما سبق ذكره فى هذا الفصل ينطبق أيضا على كل برامج عائلة (DBase) مثل (DBase IV) ، (FoxBase +) ، (FoxPro).

الفصل السادس والثلاثون

استخدام وسائل أكثر تقدما

(More Advanced Techniques)



٣٦ - ١ إستخدام الدالة (IIF)

الدالة (IIF) هى صورة أخرى للأمر (IF) وهذه الدالة يتم عن طريقها كتابة الشرط وجواب الشرط على نفس السطر حيث لاتكون هناك حاجة لكتابة الأمر (ELSE) أو الأمر (ENDIF). ويتم ذلك عن طريق كتابة الشرط وجوابه بين قوسين بعد الدالة (IIF) مع فصلهما بفاصلة (,). فمثلا الأوامر التالية تستخدم الأمر (IF).

```
IF sex = 'F'
    mname = 'Ms.' + name
ESLE
    mname = 'Mr.' + name
ENDIF
```

فعند استخدام الدالة (IIF) يتم اختصار هذه السطور إلى السطر التالى :

```
mname = IIF(sex = 'F', 'Ms.' , 'Mr.') + name
```

ويعنى هذا إختبار قيمة الحقل (sex) أى الجنس فإذا كان (F) أى (Female) أو أنثى يتم كتابة الحروف (Ms.) قبل الإسم (name) وإذا كان غير ذلك يتم كتابة الحروف (Mr.) قبل الإسم ثم يتم تخزين النتيجة فى المتغير (mname). واختصار السطور بهذه الطريقة يؤدى إلى سهولة تصحيح البرنامج بالإضافة إلى سرعة تنفيذه.

٣٦ - ٢ إستخدام ملف الخطوات (Procedure File)

عند تصميم البرنامج فإن مخطط البرامج يكتشف أن هناك بعض البرامج الفرعية (Modules) التى تستخدم بكثرة فى البرنامج. وفى كل مرة يتم فيها تنفيذ أى برنامج من هذه البرامج الفرعية يستخدم الأمر (DO) فى الإنتقال من البرنامج الرئيسى إلى البرنامج الفرعى كما سبق الإيضاح.

وحيث أن البرنامج الفرعى يكون دائما مخزنا فى ملف منفصل فإن هذا الملف المنفصل يكون موجودا على القرص حتى يتم استدعاؤه. لذلك فإن البرنامج يذهب إلى القرص فى كل مرة يراد فيها فتح ملف برنامج فرعى وهذا قد يستغرق وقتا طويلا خصوصا عند تعدد

هذه البرامج الفرعية. ولذلك يتيح برنامج (+DBase III) لمخطط البرامج استخدام مايسمى بملف الخطوات أو الملف الإجرائى (Procedure File). هذا الملف يحتوى على برامج صغيرة (Modules) تبقى فى الذاكرة المؤقتة طوال فترة تشغيل البرنامج وبالتالى فإن البرنامج لايحتاج إلى الذهاب إلى القرص عدة مرات لتحميل هذه البرامج.

والملف الإجرائى يتم إنشاؤه وتعديله باستخدام الأمر(MODIFY COMMAND) لذلك يتم إضافة الإمتداد (Prg.) إلى إسمه آليا. ويتم كتابة كل البرامج الإجرائية (Procedures) داخل هذا الملف مع ملاحظة أن كل برنامج من هذه البرامج يبدأ بالأمر (PROCEDURE) يليه إسم هذا البرنامج ثم تأتى باقى أوامر البرنامج. والسطر الأخير من كل برنامج يحتوى على الأمر (RETURN).

ملاحظة

بدلا من كتابة الملف الإجرائى (Procedure File) مباشرة يمكن كتابة كل برنامج فرعى منفصلا وذلك باستخدام الأمر (MODIFY COMMAND) يليه إسم البرنامج الفرعى المطلوب كتابته ثم يتم اختبار كل برنامج فرعى والتأكد أنه يحقق الهدف منه. وعند الإنتهاء من جميع البرامج الفرعية يتم إنشاء الملف الإجرائى (Procedure File) باستخدام الأمر (MODIFY COMMAND) أيضا. كما يتم نقل كل برنامج فرعى تم إنشاؤه إلى الملف الإجرائى عن طريق الضغط على مفتاحى (Ctrl-KR) حيث يسأل البرنامج عن إسم الملف المراد نقله إلى الملف الإجرائى. ويمكن اتباع نفس الطريقة فى إضافة أى برنامج فرعى جديد إلى الملف الإجرائى (Procedure File).

وعند الحاجة إلى تشغيل الملف الإجرائى مع البرنامج الرئيسى يستخدم الأمر (SET PROCEDURE TO) مع كتابة إسم الملف الإجرائى بعده وذلك فى بداية البرنامج ثم يتم تنفيذ أى برنامج فرعى داخل الملف الإجرائى باستخدام الأمر التالى مثلا :

```
DO Report
```

مع ملاحظة أنه لايتم فتح أكثر من ملف إجرائى واحد فى كل مرة ولكن يمكن إغلاق الملف المستخدم وفتح ملف إجرائى آخر فى أى وقت. ولكى يتم فتح ملف إجرائى آخر يستخدم الأمر (SET PROCEDURE TO) ثم كتابة إسم الملف المراد فتحه. وفى هذه الحالة يتم إغلاق الملف السابق وفتح الملف الجديد. كما يمكن إغلاق ملف إجرائى دون فتح أى ملف آخر وذلك بكتابة الأمر (CLOSE PROCEDURE) أو الأمر

(SET PROCEDURE TO) دون كتابة أى إسم بعده. ويمكن أن يحتوى الملف الإجرائى الواحد على ما لايزيد عن ٣٢ برنامجا فرعيا (Procedure).

ملاحـظة

يمكن لبرنامج إجرائى أن يستدعى برنامج إجرائى آخر فى نفس الملف الإجرائى المفتوح وذلك بكتابة الأمر (DO) وبعده إسم البرنامج الإجرائى المطلوب.

والسطور التالية توضح إنشاء ملف إجرائى الذى يتم تسميته (Test) مثلا :

```
PROCEDURE Proc1
   ? "This is a message from Proc1"
RETURN
PROCEDURE Proc2
   ? "Greetings From Proc2"
   DO Proc3
RETURN
PROCEDURE Proc3
   ? "greetings From Proc3."
RETURN
```

ولتحميل هذا الملف يتم كتابة الأمر التالى :

```
SET PROCEDURE TO Test
```

كما يمكن تشغيل البرنامج الأول والثانى كالآتى :

```
DO Proc1
DO Proc2
```

ويلاحظ فى هذه الحالة ظهور الآتى على الشاشة :

```
This is a message from Proc1.
Greetings From Proc2.
Greetings From Proc3.
```

ويلاحظ هنا ظهور الرسالة الخاصة ببرنامج إجرائى (Proc3) رغم عدم استدعائه وذلك لأن البرنامج (Proc2) يستدعيه من داخله.

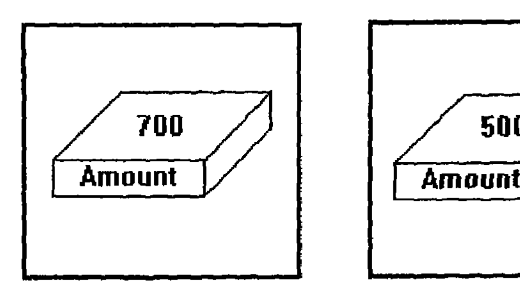
٣٦ - ٣ إخفاء المتغير العام (Public Variable)

كما سبق الإيضاح فإن المتغير العام يؤثر فى جميع البرامج حتى إذا تم إنشاؤه داخل برنامج فرعى. وذلك عكس المتغير الخاص (Private) الذى لايؤثر إلا على البرنامج الفرعى الذى تم إنشاؤه خلاله أو البرامج الفرعية المتفرعة منه. ولتوضيح ذلك نفرض أن المتغير (amount) تم إنشاؤه داخل أحد البرامج الفرعية وكانت قيمته داخل هذا البرنامج الفرعى (500) مثلا. فإذا تم استخدام المتغير (amount) فى أى برنامج آخر فإن هذا المتغير يأخذ نفس القيمة (500).

وفى بعض الأحيان يراد استخدام المتغير العام فى أحد البرامج الفرعية دون أن يتأثر بالقيمة المخزنة فى هذا المتغير أى استخدامه كمتغير خاص لهذا البرنامج الفرعى فقط. فى هذه الحالة يتم إخفاء المتغير العام عن هذا البرنامج الفرعى ويتم هذا الإخفاء باستخدام الأمر التالى :

```
PRIVATE amount
```

فعند تشغيل هذا البرنامج الفرعى يظل المتغير (amount) متغيرا خاصا بالنسبة لهذا البرنامج والبرامج الفرعية المتفرعة منه. أى أن قيمته لاتتأثر بالمتغير العام (amount) الموجود فى البرنامج الرئيسى. وعند انتهاء البرنامج الفرعى يعود المتغير (amount) إلى حالته الأولى أى يصبح متغيرا عاما.



شكل (٣٦ - ١)

ويلاحظ من الشكل (٣٦ - ١) أن المتغير (amount) يأخذ القيمة (500) فى نهاية تنفيذ البرنامج دون أن يؤثر ذلك فى قيمة نفس المتغير فى البرنامج الرئيسى.

٣٦ - ٤ إدخال المعاملات (Parameter Passing)

كما سبق الإيضاح فإن استخدام البرامج الفرعية (Modules) أو البرامج الإجرائية (Procedures) يتيح لمخطط البرامج استخدام البرنامج الفرعى عدة مرات وفى أماكن مختلفة من البرنامج ولكن فى بعض الأحيان يراد استخدام نفس البرنامج الفرعى مع تعديل بعض القيم المستخدمة به.

فمثلا قد يراد أحيانا رسم مستطيل على الشاشة حول البيانات المختلفة ولكن قد يراد رسم هذا المستطيل فى أماكن مختلفة من الشاشة. وفى هذه الحالة تسبب كتابة برنامج فرعى منفصل لكل مستطيل إستهلاكا للوقت والذاكرة المتاحة. لذلك يتيح البرنامج إستخدام المعاملات (Parameters) التى يتم إدخالها إلى البرنامج الفرعى من البرنامج الذى قام باستدعائه. ويتم تنفيذ ذلك عن طريق كتابة البرنامج مع استخدام متغيرات تمثل القيم المطلوب استخدامها كمعاملات. فمثلا يمكن كتابة برنامج إجرائى (Procedure) الذى يتم تسميته (Box) كالآتى :

```
PARAMETERS beginrow, begincol, endrow, endcol
CLEAR
@ beginrow, begincol TO endrow, endcol DOUBLE
RETURN
```

ويلاحظ فى هذا البرنامج كتابة الأمر (PARAMETERS) يليه أسماء المعاملات المتغيرة (Beginrow) ، (Begincol) ، (Endrow) ، (Endcol).

وعندما يراد رسم هذا المستطيل فى مكان محدد على الشاشة مثلا يستخدم الأمر التالى :

```
DO Box WITH 10,9,23,75
```

ويؤدى هذا إلى إستدعاء البرنامج الفرعى (Box) مع إدخال المعاملات المطلوبة مكان

المعاملات المتغيرة (beginrow) ، (begincol) ، (endrow) ، (endcol) بنفس ترتيبها. ويمكن استخدام هذا البرنامج (Box) فى أى مكان آخر من البرنامج مع تغيير هذه المعاملات.

٣٦ - ٥ إستخدام الأمر (RUN)

يمكن تشغيل برامج أخرى من خلال البرنامج مثل برامج نظام التشغيل الساكنة فى الذاكرة (Memory Resident Programs) وذلك باستخدام الأمر(RUN). كما يمكن استخدام الأمر (!) بدلا من الأمر (RUN) حيث أنه يؤدى نفس الغرض. وهذه البرامج يتم تشغيلها بكتابة إسم البرنامج مثل الأمر (FORMAT) الذى يستخدم فى تجهيز قرص جديد والأمر (CHKDSK) الذى يستخدم فى اختبار القرص واكتشاف أى قطاعات غير سليمة (Bad Sectors). ويجب ملاحظة أن الأمر (RUN) يتطلب ذاكرة مؤقتة (RAM) تزيد عن ٢٥٦ كيلوبايت. كما يجب أن يكون الملف (Command.com) موجودا فى الفهرس الرئيسى (Root Directory).

ويستخدم الأمر (RUN) بصفة خاصة فى إدخال تاريخ اليوم الحالى إلى البرنامج. ويتم ذلك عن طريق إدخال تاريخ اليوم فى متغير ذاكرة ثم استخدام الأمر (DATE) فى إدخال هذا التاريخ.

```
Today = '7/5/89'
RUN DATE & Today
```

ويؤدى هذا إلى إدخال التاريخ الموجود فى متغير الذاكرة (Today) لكى يصبح هو تاريخ اليوم الحالى.

٣٦ - ٦ نظــام التشغيــل

عند كتابة برنامج ويراد زيادة كفاءة وإنتقالية هذا البرنامج (Portability) أى قدرته على العمل على نظم تشغيل متعددة فإن ذلك يتطلب أن يعرف البرنامج نظام التشغيل الذى يعمل عليه حتى يتم إدخال تجهيز معين للبرنامج حتى يعمل على هذا النظام. ويمكن لمخطط البرامج استخدام الدالة (OS()) فى تنفيذ ذلك وهذه الدالة عند كتابتها فى البرنامج فإنها تعطى سلسلة حرفية (String) تمثل إسم هذا النظام. ويمكن ملاحظة ذلك من خلال البرنامج التالى :

```
STORE OS() TO opsys
IF SUBSTR(opsys, 1, 4) = 'UNIX'
DO Setunix
ENDIF
```

وهذا البرنامج يقوم أولا باختبار نظام التشغيل المستخدم فإذا وجد أنه النظام (UNIX) فإنه ينفذ البرنامج (Setunix) الذى يؤدى إلى تجهيز البرنامج للعمل على نظام التشغيل يونيكس.

٣٦ - ٧ التعويض بالماكرو (Macro Substitution)

تستخدم الدالة (&) للتعويض عن القيمة الحرفية لمتغير حرفى فى أى مكان داخل البرنامج. وهذه الدالة مهمة جدا لزيادة كفاءة البرنامج وسرعته وهى تكون ضرورية مع بعض الأوامر مثل الأمر (FIND) مثلا. ولتوضيح وظيفة هذه الدالة يمكن دراسة الأوامر التالية :

```
STORE "Tarek" TO mname
USE Cadets INDEX Name
FIND & mname
```

فى هذه الحالة يقوم البرنامج بالبحث عن السلسلة الحرفية (Tarek) المخزنة فى المتغير (mname) كما يمكن إدخال الماكرو داخل سلسلة حرفية أخرى كما يتضح من الأوامر التالية :

```
STORE "Hasan" TO mname
STORE "Hello & mname" TO greeting
? greeting
```

وعند الضغط على مفتاح الإدخال يظهر الآتى :

```
Hello Hasan
```

كما يمكن إضافة أى حروف أخرى إلى الماكرو وذلك بكتابة النقطة (.) وبعدها هذه الحروف كالآتى مثلا :

```
STORE "Hasan" TO mname
STORE "Hello & mname . ein" TO greeting
? greeting
```

وعند الضغط على مفتاح الإدخال يظهر الآتى :

```
Hello Hasanein
```

ويلاحظ هنا أنه تم إضافة الحروف (ein) إلى الإسم (Hasan).

٣٦ - ٨ التحكم فى الألوان (Colors)

يمكن الحصول على الألوان المطلوب ظهورها أثناء تنفيذ البرنامج. ويتم ذلك عن طريق كتابة الأمر (SET) عند مشيرة النقطة (Dot Prompt) يلاحظ فى هذه الحالة ظهور قوائم التجهيز. وعن طريق تحريك المؤشر الضوئى (Highlight) أعلى الشاشة إلى الإختيار (Screen) والضغط على مفتاح الإدخال تظهر قائمة بالألوان المطلوب إختيارها وذلك بالنسبة للكتابة (Foreground) والخلفية (Background).

ويلاحظ أن هناك قائمة خاصة بالشاشة (Standard Display) وقائمة أخرى خاصة بالأعمدة الضوئية التى تظهر خلال الشاشة (Enhanced Display). كما أن هناك قائمة خاصة بالحدود (Borders) التى تظهر حول الشاشة. كما يلاحظ وجود إختيار لشدة اللون (Intensity) فى جميع الحالات. وشدة اللون تعنى إذا كان مضيئا (Bright) أو معتما (Dim). كما أن هناك اختيار اللون المتلألىء (Blinking) أو غير المتلألىء. انظر الشكل (٣٦ - ٢).

وهذه الألوان تعطى شاشات الإدخال وشاشات عرض البيانات شكلا جذابا ومثيرا. وقد يريد مخطط البرامج فى بعض الأحيان التحكم فى هذه الألوان من خلال البرنامج حيث أن ذلك يتيح له تنبيه المستخدم إلى أهمية بعض المعلومات وإعطاء التأثير المطلوب. ويتم ذلك من خلال الأمر (SET COLOR ON) للتحويل من الشاشة أحادية اللون (Monochrome Monitor) إلى الشاشة الملونة (Colored Monitor) ثم يتم اختيار

الألوان بواسطة الأمر (SET COLOR TO) ثم كتابة سلسلة حرفية (String) تمثل الألوان المطلوبة. وهذه السلسلة تنقسم إلى أربعة أجزاء، الجزء الأول يمثل لون الشاشة، والجزء الثانى يمثل لون الأعمدة الضوئية (Enhanced)، والجزء الثالث يمثل الحدود (Borders) والجزء الرابع يمثل الخلفية (Background). ويتم تمثيل كل لون بحرف أو حرفين حسب الجدول المبين.

```
Options Screen   Keys   Disk   Files Margin  Decimals
        Display Type              EColor25
        Standard Display
             Foreground:          Magenta
             Background:          White
             Intensity:           Dim
             Blink:               No
             Blank:               No
        Enhanced Display
             Foreground:          Green
             Background:          White
             Intensity:           Dim        Standard Display
             Blink:               No         Enhanced Display
             Blank:               No
SET              ||<C:>||                    ||Opt: 3/11
 Positionselection bar - ↑↓.  Change - ◄─┘.  Leave menu-←→
    Choose foreground and background color or intensity.
```

شكل (٣٦ - ٢)

اللون	الحروف
أحمر	R
بنفسجى	RB
بنى	GR
أبيض	W
أسود	N
أزرق	B
أخضر	G
سماوى	BG
خالى	X

ولتوضيح ذلك يمكن كتابة الأمر التالى :

SET COLOR TO GR+/B , W/R, GR

وهذا يؤدى إلى تحديد اللون الأصفر للكتابة فى الشاشة على خلفية زرقاء كما يؤدى إلى تحديد لون الأعمدة الضوئية (Enhanced Video) ليكون لون الكتابة أبيض على خلفية حمراء كما يحدد لون الحدود (Borders) ليكون بنيا.

ويلاحظ إضافة علامة (+) مع الحروف (GR) لتحويل اللون من بنى إلى أصفر حيث أن علامة (+) تؤدى إلى زيادة شدة اللون (Intensity). كما يمكن إضافة العلامة (*) إلى أى لون لتحويل اللون إلى الوضع المتلألىء (Blinking). ويستخدم الحرف (X) فى بعض الحالات ليجعل الكتابة بنفس لون الخلفية. ويفيد ذلك عند كتابة البرامج التى تستخدم فى إدخال كلمة مرور للبرنامج (Password) حيث يتطلب ذلك كتابة كلمة المرور بحيث لاتظهر أمام أى شخص موجود أمام الحاسب لذلك يتم كتابتها بنفس لون الخلفية.

٣٦ - ٩ إستخدام الإختصارات فى كتابة الأوامر

يتيح برنامج (+DBase III) لمخطط البرامج كتابة الأوامر بطريقة مختصرة تساعد على توفير الوقت والجهد المستهلك فى كتابة البرنامج حيث يمكن كتابة الأربعة حروف الأولى فقط من كل أمر وذلك كالآتى مثلا :

MODI COMM

وذلك بدلا من الأمر (MODIFY COMMAND). وينطبق هذا على أى أمر من أوامر برنامج (+DBase III).

ملاحظة

ما سبق ذكره فى هذا الفصل ينطبق أيضا على كل برامج عائلة (DBase) مثل (DBase IV) ، (+ FoxBase) ، (FoxPro).

الفصل السابع والثلاثون

اختبار وتصحيح البرنامج

(Testing and Debugging)



هناك أنواع متعددة من الأخطاء التى يمكن أن تظهر فى البرنامج يمكن تصنيفها إلى الأخطاء الهجائية (Misspelling) وأخطاء القواعد (Syntax Errors) والأخطاء المنطقية (Logical Errors). والأخطاء الهجائية يمكن اكتشافها بسهولة عن طريق مراجعة قائمة الأوامر (List) واختبار هجاء كل كلمة للوصول إلى هذه الأخطاء وتصحيحها. وكذلك فإن أخطاء القواعد (Syntax Errors) تتطلب مراجعة كل أمر ومقارنته بالشكل الخاص به (Syntax) والذى تم وضعه بواسطة برنامج (DBase III+) أو برامج عائلة (DBase) الأخرى مثل(DBase IV) ، (FoxBase +) ، (FoxPro). وللرجوع إلى الشكل (Syntax) الخاص بكل أمر يمكن استخدام شاشات المساعدة (Help Screens) التى توضح الشكل الخاص بكل أمر مع شرح كل الإختيارات الممكنة. أما الأخطاء المنطقية فإنها تعتبر أعقد الأخطاء وأصعبها من حيث الإكتشاف أو التصحيح (Debugging) حيث أنها أخطاء تتعلق بالتسلسل المنطقى للبرنامج وتسمى أحيانا أخطاء وقت التشغيل (Run Time Errors) لأنها لاتظهر إلا عند التشغيل الفعلى للبرنامج. وبعض هذه الأخطاء لايؤدى إلى توقف البرنامج ولكنه يؤدى إلى الحصول على نتائج غير سليمة للبرنامج.

٣٧ - ١ خطوات الإختبار

كما سبق الإيضاح فإن تصميم البرنامج يعتمد على الطريقة التركيبية (Structured) حيث يتم تقسيم البرنامج إلى برامج فرعية (Modules) كل منها يؤدى مهمة محددة (Task). وهذه الطريقة تسهل كتابة البرنامج إلى درجة كبيرة حيث أن كتابة البرنامج الفرعى الذى يؤدى مهمة محددة أسهل كثيرا من كتابة برنامج كبير يؤدى وظائف متعددة. كما يمكن فى البرامج الكبيرة توزيع البرنامج على مجموعة من مخططى البرامج بحيث يقوم كل منهم بكتابة برنامج فرعى محدد. وكما تسهل هذه الطريقة كتابة البرامج فإنها أيضا تسهل اختبارها وتصحيحها حيث يتم اختبار كل برنامج فرعى وتصحيحه مستقلا عن البرامج الأخرى.

وهناك خطوات قياسية (Standard) لاختبار وتصحيح أى برنامج يمكن تلخيصها فيما يلى :

١ - يتم كتابة كل برنامج فرعى (Module) وتوثيقه ثم اختباره بمجرد الإنتهاء من كتابته ثم الإنتقال إلى البرنامج الفرعى التالى وكتابته ثم اختباره وتصحيحه وهكذا.

٢ - عند الإنتهاء من مجموعة من البرامج التى تكون برنامجا مركبا (Composite) يتم ربطها معا وتشغيلها واختبار هذا البرنامج الجديد.

٣ - يتم تجميع البرامج المركبة فى برامج أكبر عن طريق إضافة البرامج الفرعية الجديدة (Modules) التى يتم اختبارها منفصلة.

٤ - عندما يتم تجميع البرنامج الكبير يتم اختباره أيضا بنفس الطريقة.

٥ - يتم اختبار البرنامج بواسطة أشخاص آخرين غير مشتركين فى كتابة البرامج الفرعية وهذا الإختبار يسمى اختبار ألفا (Alpha Testing). وهو يتم عادة قبل البدء الفعلى فى استخدام البرنامج ويتم خلال هذا الإختبار تصحيح أى أخطاء تظهر فى البرنامج.

٦ - يتم توزيع البرنامج على مجموعة محددة من المستخدمين (Users) ويقوم هؤلاء المستخدمون باختباره .وهذا الاختبار يسمى اختبار بيتا (Bita Testing).

٧ - بعد ذلك يصبح البرنامج جاهزا للاستخدام بواسطة مستخدمين آخرين ولكن هذا لاينهى مرحلة الإختبار والتصحيح (Debugging) لأن هذه المرحلة تظل مستمرة ربما لعدة شهور أو عدة سنوات فى بعض البرامج الكبيرة.

٣٧ - ٢ أوامر التصحيح (Debugging Commands)

يوفر البرنامج مجموعة من الأوامر التى تساعد مخطط البرامج على اكتشاف الأخطاء وتصحيحها. ويتوقف إختيار أى أمر من هذه الأوامر على طبيعة الخطأ المتوقع والطريقة المطلوبة لاكتشافه وتصحيحه. حيث أن بعض هذه الأوامر يؤدى إلى تعليق تنفيذ أوامر البرنامج (Suspend) وبعضها يؤدى إلى تنفيذ البرنامج خطوة خطوة حتى يتم اكتشاف الخطوة التى تسبب حدوث الخطأ.

٣٧ - ٣ تعليق تنفيذ البرنامج (Suspend)

عندما يحدث أى خطأ أثناء تشغيل البرنامج فإن البرنامج يتوقف وتظهر الرسالة التالية :

Cancel , Ignore , Suspend (C, I, S)

واختيار الحرف (C) أى (CANCEL) يؤدى إلى إنهاء البرنامج والعودة إلى مشيرة النقطة (Dot Prompt).

واختيار الحرف (I) أى (IGNORE) يؤدى إلى إكمال تنفيذ البرنامج وتخطى النقطة التى ظهر عندها الخطأ.

أما اختيار الحرف (S) أى (SUSPEND) فإنه يؤدى إلى توقف البرنامج مؤقتا والعودة إلى مشيرة النقطة (Dot Prompt). ومن هذا الوضع يمكن بحث الأخطاء المحتملة وتصحيحها من خلال مشيرة النقطة وذلك عن طريق العودة إلى قائمة البرنامج وتصحيح الأمر المتوقع حدوث الخطأ منه. وخلال هذه العملية يكون البرنامج معلقا أى أن تشغيله متوقف حتى يقوم مخطط البرامج بكتابة الأمر (RESUME) والضغط على مفتاح الإدخال حيث يستمر تنفيذ البرنامج مرة ثانية.

٣٧ - ٤ إستخدام مخزن التاريخ (History)

ويقصد بمخزن التاريخ هنا مخزن ذاكرة مؤقت (Buffer) يتم فيه تخزين آخر عشرين أمرا تم إدخالها من خلال مشيرة النقطة. وفى أى وقت يراد رؤية الأوامر التى سبق إدخالها يستخدم مفتاح السهم لأعلى (↑) حيث أن كل ضغطة عليه تظهر الأمر السابق مباشرة وذلك بحد أقصى عشرين أمرا. ويتيح ذلك أيضا تنفيذ بعض الأوامر التى سبق إدخالها دون الحاجة إلى كتابتها من جديد حيث يكفى فى هذه الحالة الضغط على مفتاح (↑) عدة مرات حتى يظهر الأمر المطلوب ثم الضغط على مفتاح الإدخال.

ويمكن زيادة الأوامر التى يمكن تخزينها فى هذا المخزن (History) عن طريق كتابة الأمر (SET HISTORY TO) ثم كتابة عدد الأوامر المطلوب تخزينها. فمثلا الأمر التالى :

```
SET HISTORY TO 30
```

يسمح بتخزين ٣٠ أمرا فى مخزن التاريخ وهذا يؤدى إلى الإحتفاظ بآخر ٣٠ أمرا تم إدخالها. ويمكن عرض هذه الأوامر كما سبق الإيضاح باستخدام السهم (↑) وباستخدام الأمر (DISPLAY HISTORY) أو الأمر (LIST HISTORY).

هذا مايحدث بالنسبة للأوامر التى يتم إدخالها من خلال مشيرة النقطة (Dot Prompt). أما أوامر البرنامج فإنها فى الوضع المبدئى للبرنامج (Default) لايتم تخزينها فى مخزن التاريخ (History) ولكن يمكن لمخطط البرامج تغيير هذا

الوضع المبدئى عن طريق كتابة الأمر (SET DOHISTORY ON) وذلك قبل تشغيل البرنامج. وعند توقف البرنامج نتيجة وجود أى خطأ (Error) يمكن لمخطط البرامج عرض آخر عشرين أمرا تم إدخالها وذلك بكتابة الأمر التالى :

DISPLAY HISTORY

أو الأمر التالى :

LIST HISTORY

كما يمكن زيادة عدد هذه الأوامر باستخدام الأمر (SET HISTORY TO) كما سبق الإيضاح. ويجب ملاحظة أن استخدام الأمر (SET DOHISTORY ON) سوف يؤثر على كفاءة وسرعة تنفيذ البرنامج. لذلك يجب التأكد من إعادة هذا الأمر بعد اختبار البرنامج إلى الوضع المبدئى (Default) كالآتى :

SET DOHISTORY OFF

٣٧ - ٥ مراقبة تنفيذ البرنامج

يتيح برنامج (DBase III+) لمخطط البرامج استخدام بعض الأوامر التى تساعده على مراقبة خطوات تنفيذ البرنامج. وهذا يساعده على اكتشاف مكان الخطأ وأحيانا التعرف على سبب هذا الخطأ. ويتم ذلك باستخدام الأمر (SET TALK ON) والأمر (SET ECHO ON) والأمر (SET DEBUG ON).

٣٧ - ٦ الأمر (SET TALK ON)

الوضع المبدئى لهذا الأمر يكون (ON) وهذا يعنى ظهور رسائل توضح تنفيذ كل أمر. ولكن عند كتابة البرنامج يتم تغيير هذا الوضع المبدئى عن طريق كتابة الأمر (SET TALK OFF) وذلك حتى لاتظهر هذه الرسائل للمستخدم عند تنفيذ البرنامج. ولذلك فعند تصحيح البرنامج (Debugging) يفضل إعادة الوضع المبدئى مرة ثانية حتى تظهر هذه الرسائل وذلك لأنها تفيد عند حدوث خطأ فى نقطة معينة من البرنامج حيث توضح هذه الرسائل أحيانا سبب الخطأ.

٣٧ - ٧ الأمـــــر (SET ECHO ON)

هذا الأمر يعتبر صورة مكبرة من الأمر (SET TALK ON) حيث أنه يؤدى إلى ظهور خطوات تنفيذ كل أمر داخل الحاسب. وهذا يؤدى إلى ظهور كل شيء ينفذه البرنامج على الشاشة أثناء تشغيله. وعند توقف البرنامج نتيجة خطأ معين (Error) يكون من السهل الوصول إلى سبب هذا الخطأ.

٣٧ - ٨ الأمـــــر (SET STEP ON)

يستخدم هذا الأمر لعرض خطوات تنفيذ البرنامج مثل الأمر (SET ECHO ON) تماما ولكنه يختلف عنه فى أنه يعرض هذه الخطوات خطوة خطوة. ويتوقف التنفيذ بعد كل خطوة حتى يضغط مخطط البرامج على أى مفتاح للإنتقال إلى الخطوة التالية. وهذه الطريقة كما هو واضح تؤدى إلى إبطاء تنفيذ البرنامج بدرجة كبيرة ولكنها تفيد فى إكتشاف الأخطاء الدقيقة جدا التى يصعب اكتشافها باستخدام الوسائل السابقة.

٣٧ - ٩ الأمـــــر (SET DEBUG ON)

يستخدم هذا الأمر أيضا فى اكتشاف الأخطاء الدقيقة التى يصعب اكتشافها بالطرق السابقة وهو يؤدى عند كتابته من مشيرة النقطة (Dot Prompt) إلى توجيه مخرجات الأمر (SET ECHO ON) أو مخرجات الأمر (SET STEP ON) إلى الطابعة بدلا من عرضها على الشاشة. وهذا يؤدى إلى الحصول على نسخة مطبوعة (Hard Copy) من أوامر البرنامج متضمنة خطوات تنفيذ كل أمر كما تحدث داخل الحاسب.

٣٧ - ١٠ عرض محتويات الذاكرة (Display Memory)

عند ظهور رسالة خطأ مثل(Variable Not Found) أو "Data Type Mismatch" فربما يكون سبب هذا الخطأ عدم إنشاء متغير ذاكرة أو استخدام نوعين من البيانات فى سطر واحد (بيانات حرفية مع بيانات عددية مثلا). فى هذه الحالة يمكن تعليق البرنامج (Suspend) ثم عرض محتويات الذاكرة باستخدام الأمر (DISPLAY MEMORY). حيث يتم فى هذه الحالة عرض جميع المتغيرات الموجودة فى الذاكرة فى هذه اللحظة مع تحديد حالة هذه المتغيرات.

ويمكن إيقاف البرنامج فى أى وقت بالضغط على مفتاح الهروب (ESC) حيث تظهر الرسالة التالية :

Cancel , Ignore, Suspend (C,I,S)

ثم يتم اختيار (S) لتعليق البرنامج وعرض محتويات الذاكرة كما سبق الإيضاح.

كما يمكن كتابة الأمر (SUSPEND) داخل البرنامج فى المنطقة المشكوك فى وجود خطأ فيها. وعند توقف البرنامج يتم كتابة الأمر (DISPLAY MEMORY) الذى يؤدى إلى عرض محتويات الذاكرة باستخدام الأمر (DISPLAY MEMORY) كما سبق الإيضاح.

٣٧ - ١١ عـرض الحـــالـة (Display Status)

هذا الأمر يؤدى إلى عرض حالة البيئة (Environment) الخاصة بالبرنامج. وهذا يشمل عرض أسماء ملفات قواعد البيانات المفتوحة وملفات الفهرس الخاصة بها بالإضافة إلى البيانات المبدئية لبرنامج (DBase III+) أو برامج عائلة (DBase) الأخرى مثل (DBase IV) ، (FoxBase +) ، (FoxPro) حسب البرنامج المستخدم.

٣٧ - ١٢ عرض تركيب ملف قاعدة البيانات

فى بعض الأحيان يكون هناك أخطاء ناتجة عن اختيار حقول بيانات غير مطابقة للمتغيرات المستخدمة فى البرنامج. ويمكن إكتشاف هذه الأخطاء عن طريق عرض تركيب الملف بواسطة الأمر (DISPLAY STRUCTURE) حيث يتم عرض حقول البيانات على الشاشة كالآتى مثلا :

Field	Field Name	Type	Width	Dec
1	Name	Character	30	
2	Address	Character	30	
3	Phone	Character	10	

كما يمكن طباعة هذا التركيب على الطابعة عن طريق كتابة الأمر التالى :

DISPLAY STRUCTURE TO PRINT

ملاحظة

ما سبق ذكره فى هذا الباب ينطبق أيضا على كل برامج عائلة (DBase) مثل (DBase IV) ، (FoxBase +) ، (FoxPro).





الجزء الثالث

أوامر ودوال برامج (DBase)

(FoxPro, DBaseIV, DBase III+)



هذا الجزء يعتبر إستكمالا للجزء الثانى ويتيح للقارىء التعرف على أوامر ودوال برامج عائلة (DBase) مع الشرح التفصيلى لكل منها. ورغم أن الجزء الثانى تضمن شرح معظم هذه الدوال إلا أن مؤسسة " دلـتـــا " رأت إضافة هذا الجزء حتى يصبح المرجع شاملا يغنى القارىء عن الإطلاع على أى مراجع أخرى فى هذا الموضوع. ومن خلال هذا الجزء يستطيع القارىء الوصول إلى الأمر المطلوب حيث أن الأوامر والداول مرتبة حسب الترتيب الهجائى للحروف. وسوف يجد القارىء الشرح الوافى لكل أمر متضمنا الرسم التوضيحى الملائم والأمثلة المناسبة.



الفصل الثامن والثلاثون

أهم الأوامـــر المستخـدمـــة



إن قائمة الأوامر المستخدمة فى كتابة البرامج عن طريق برنامج (+DBase III) وكذلك باقى برامج عائلة (DBase) مثل (Dbase IV) ، (+ FoxBase) ، (FoxPro) كبيرة جدا ولن يتسع المجال لدراستها بالتفصيل فى هذا الكتاب. ولكن سيتم فى هذا الفصل إلقاء الضوء على معظم هذه الأوامر مع دراسة تفصيلية لها كلما أمكن. مع ملاحظة أن الأوامر مرتبة حسب الترتيب الهجائى للحروف الإنجليزية.

ملاحظة

القيم الموجودة داخل قوسين مربعين [ ] هى قيم إختيارية يستطيع المستخدم كتابتها أو عدم كتابتها. كذلك فإن مايكتب بين الزاويتين < > يعنى عناصر مختارة بواسطة المستخدم نفسه أما الزاويتين نفسهما فلا يكتبا ضمن العنصر.

١ - الأمر (?)

يستخدم هذا الأمر لعرض محتويات متغير ذاكرة أو حقل معين فى ملف قاعدة البيانات عن طريق كتابة إسم هذا المتغير أو الحقل بعد علامة الإستفهام وهو يعنى استفهام أو سؤال عن قيمة مطلوبة. وتتم الكتابة من أول السطر التالى.

والصورة العامة للأمر كالآتى :

```
? [< expression list >]
```

حيث (Expression List) هو إسم متغير ذاكرة أو حقل أو أى علاقة مطلوب حساب قيمتها.

وعند كتابة هذا الأمر بدون أى علاقة بعده فإن هذا يؤدى إلى عرض سطر خال. وتستخدم هذه العملية عندما يراد عرض سطور خالية بين المخرجات المطبوعة.

أمثلة

لحساب نتيجة معادلة حسابية يتم كتابة السطر التالى :

```
? 2 * 2 + (8/2)
8
```

ولعرض تاريخ اليوم الحالى يتم كتابة السطر التالى :

```
. ? DATE()
01/07/90
```

ولعرض بيانات حقول فى قاعدة البيانات يتم كتابة السطور التالية :

```
. USE Cadets
. ? name , address
Mohamed Aly       12 - Nasr City - Cairo
```

٢ - الأمــــر (??)

هذا الأمر يؤدى نفس العمل مثل الأمر السابق تماما ولكنه يسمح بالكتابة من أى مكان فى السطر ولايشترط الكتابة من أول السطر.

٣ - الأمــــر (@)

هذا الأمر يستخدم فى إنشاء شاشات الإدخال والتقارير بالإضافة إلى التحكم فى مكان عرض البيانات على الشاشة. وهناك عدة صور لهذا الأمر يتم دراسة كل منها على حدة. وهى :

أ - الصورة الأولى

```
@ <row, col> SAY <expression list>
```

وتستخدم هذه الصورة فى عرض أى بيانات موجودة فى متغيرات ذاكرة أو حقول داخل ملف قاعدة البيانات أو رسائل خاصة يراد عرضها على المستخدم. ويتم ذلك عن طريق كتابة الإحداثيات المطلوب عرض البيانات بدءا منها وهذه الإحداثيات هى رقم السطر (Row) ورقم العمود (Column).

مـثـال

```
USE Cadets
@ 10,10 SAY name
```

وعند تنفيذ هذه الأوامر يظهر السطر التالى مثلا :

Ahmed Salem

وذلك بدءا من السطر العاشر والعمود العاشر.

ملاحـظة

الرقم المقابل للسطر (row) يأخذ أى قيمة من صفر إلى ٢٣ والرقم المقابل للعمود (col) يأخذ أى قيمة من صفر إلى ٧٩ مع ملاحظة أن ترقيم السطور يبدأ من أعلى وترقيم الأعمدة يبدأ من اليسار. ويمكن تحويل السطور المكتوبة بواسطة هذا الأمر إلى الطابعة عن طريق كتابة الأمر (SET DEVICE TO PRINT).

ب - الصورة الثانية

@ <row,col> SAY <expression list> PICTURE <clause>

وتستخدم هذه الصورة كالصورة السابقة تماما مع إضافة التحكم فى شكل البيانات المعروضة عن طريق الأمر (PICTURE). ويلى هذه الكلمة تعبير (Clause) يحدد صورة هذه البيانات ويتم استخدام نموذج (Template) مكان هذا التعبير (Clause). وهذا النموذج قد يكون مجموعة من الرموز (Symbols) يمثل كل منها حرفا من حروف البيانات التى تظهر فى هذا السطر. وقد تكون دالة معينة (Function) يتم إدخالها للتحكم فى شكل جميع الحروف مرة واحدة.

رموز النموذج (Template Symbols)

وهى رموز يستخدم بعضها مع الأمر (@...SAY) ولكنها تستخدم فى الغالب مع الأمر (@...GET) كما سيتم الإيضاح. والجدول التالى يوضح هذه

الرموز ومعنى كل منها :

9 ويسمح بظهور الأعداد فقط.
* ويسمح بظهور الأعداد وعلامات الجمع والطرح والمسافات بين الأرقام.
A ويسمح فقط بظهور الحروف الهجائية.
L ويسمح بظهور البيانات المنطقية.
Y ويسمح بظهور البيانات المنطقية Y , y , N , n .
N ويسمح بكتابة الحروف أو الأرقام.
X ويسمح بكتابة أى حروف أو حروف خاصة.
! وهو يحول الحروف الصغيرة الى كبيرة (Capital).
$ وهو يعرض علامة الدولار قبل العدد.
. وهو يحدد مكان العلامة العشرية (Decimal Point).

دوال النموذج (Template Functions)

كما سبق الإيضاح فإن هذه الدوال تتحكم فى شكل الحروف بالكامل ولايلزم كتابة دالة لكل حرف. ويمكن الجمع بين الدالة (Function) والرموز (Symbols) فى نفس التعبير بعد الأمر (PICTURE) على أن يتم كتابة الدالة أولا ثم الرموز. وتبدأ الدالة عادة بالحرف @ كما يمكن كتابة كلمة (FUNCTION) بدلا من الحرف @ حتى لايحدث خلط بينه وبين الأمر @ الموجود فى أول السطر.

والجدول التالى يوضح كل دالة ومعنى كل منها :

C وهى تعرض الحروف(CR) أى (Credit) بعد الأعداد السالبة.
(وهى تعرض الأعداد السالبة بين قوسين.
B وهى تؤدى إلى ضبط الأعداد من اليسار.
Z وهى تؤدى إلى حذف الأرقام التى قيمتها صفر.
D وهى تؤدى إلى عرض التواريخ بالصورة الأمريكية.
E وهى تؤدى إلى عرض التواريخ بالصورة الأوربية.
A وهى تؤدى إلى عرض الحروف الهجائية فقط.
! وهى تؤدى إلى تحويل الحروف إلى حروف كبيرة (Capital).
R وهى تؤدى إلى عرض حروف خاصة بين البيانات المعروضة.

S<n> وهى تؤدى إلى تحديد عرض البيانات المعروضة بعدد (n) من الحروف. وتسمح بزحزحة الحروف (Scrolling) خلال هذا العرض.

أمثلـــة

السطر التالى يؤدى إلى ضبط العدد الموجود فى الحقل (amount) جهة اليسار (Left Justified) وذلك عن طريق الدالة (B).

```
@ 5,5 SAY amount PICTURE '@B 9,999,999.99'
```

كما يمكن كتابة نفس السطر السابق بصورة أخرى كالآتى :

```
@ 5,5 SAY amount FUNCTION 'B';
 PICTURE '9,999,999.99
```

ويمكن كتابة عدة دوال مع نفس الصورة (PICTURE) كالآتى :

```
@ 5,5 SAY amount PICTURE '@ XC 999.99'
```

جـ - الصورة الثالثة

```
@<row,col> GET <variable> PICTURE <clause>
     RANGE <exp>,<exp>
```

وتستخدم هذه الصورة فى عرض عمود ضوئى على الشاشة يمثل المتغير (variable) المكتوب. ومن خلال هذا العمود الضوئى يستطيع المستخدم إدخال البيانات المطلوبة والتى يتم تخزينها فى المتغير. ويجب ملاحظة أن هذا المتغير يلزم إنشاؤه أولا قبل كتابة هذا الأمر. وكلمة (PICTURE) تم شرحها فى الصورة السابقة وهى تؤدى هنا إلى التحكم فى شكل البيانات التى يدخلها المستخدم. كما تؤدى إلى تحويلها إلى الشكل المطلوب إدخاله فى المتغير. وكلمة (RANGE) تستخدم مع المدخلات العددية والتاريخية لتحديد أقل قيمة وأكبر قيمة مطلوب إدخالها فى المتغير.

ولتحديد مدى تاريخى مثلا يجب أولا تحويل التاريخ من الحروف إلى تاريخ عن طريق الدالة (CTOD). فمثلا يمكن كتابة المدى بين تاريخين كالآتى :

```
RANGE CTOD('01/01/90'), CTOD('02/05/90')
```

وهذا يحدد المدى من ١/١/٩٠ إلى ٥/٢/٩٠

ويمكن كتابة الحد الأدنى فقط أو الحد الأقصى فقط كالآتى مثلا : (Range 30,). وهذا يعنى أى عدد يزيد عن (٣٠) أما (Range , 100) فهو يعنى أى عدد أقل من (١٠٠).

د - الصورة الرابعة

```
@ <row1, col1> TO <row2, col2> [ DOUBLE ]
```

وتستخدم هذه الصورة فى رسم مستطيل يبدأ من النقطة الممثلة بالإحداثيات <row1, col1> التى تمثل أعلى نقطة يسار هذا المستطيل وينتهى بالنقطة الممثلة بالاحداثيات < row2, col2> التى تمثل أدنى نقطة يمين المستطيل. واستخدام (DOUBLE) إختيارى وهو يؤدى إلى رسم المستطيل بخطوط مزدوجة. وتفيد هذه الصورة فى رسم أشكال هندسية فى شاشة إدخال البيانات تعطى الشاشة شكلا مثيرا وجذابا.

أمثلة

- السطر التالى يؤدى إلى رسم مستطيل بخطوط مفردة (Single).

```
@ 2,20 TO 8,60
```

- السطر التالى يؤدى إلى رسم مستطيل بخطوط مزدوجة (Double).

```
@ 1,10 TO 7,50 DOUBLE
```

- السطر التالى يؤدى إلى رسم خط أفقى.

@ 3,5 TO 3,30

وذلك لأن رقم السطر ثابت فى النقطتين.

- السطر التالى يؤدى إلى رسم خط رأسى.

@ 1,20 TO 20,20

وذلك لأن رقم العمود ثابت فى النقطتين.

هـ - الصورة الخامسة

@ <row1, col1> CLEAR TO <row2, col2>

وتستخدم هذه الصورة فى مسح مستطيل من الشاشة يبدأ من النقطة < row1, col1 > وينتهى بالنقطة <row2, col2 >. ويمكن استخدام هذه الصورة فى مسح أجزاء مختلفة من الشاشة حسب الحاجة. كما يمكن مسح سطر واحد عن طريق كتابة أول نقطة فى هذا السطر وآخر نقطة فيه.

مـثال

@ 1,0 CLEAR TO 6,50

وهذا الأمر يؤدى إلى مسح مستطيل يبدأ من النقطة (1 , 0) وينتهى بالنقطة (6 , 50).

٤ - الأمـــر (ACCEPT)

يستخدم هذا الأمر فى عرض رسالة للمستخدم واستقبال قيمة معينة يدخلها المستخدم ردا على هذه الرسالة. حيث يتم تخزين هذه القيمة فى متغير ذاكرة يتم إنشاؤه من خلال هذا الأمر.

والصورة العامة لهذا الأمر كالآتى :

ACCEPT [<message>] TO <memvar>

ويجب ملاحظة أن الرسالة (message) فى هذه الحالة يمكن أن تكون متغير ذاكرة حرفى (Character) أو تكون سلسلة حرفية يتم كتابتها بين علامات تنصيص (Quotation). كما يجب ملاحظة أن المتغير (memvar) يكون دائما متغيرا حرفيا (Character).

مثـــال

ACCEPT "Enter your name:" TO mname

فى هذه الحالة تظهر الرسالة التالية للمستخدم :

Enter your name :

وعندما يكتب المستخدم إسمه ويضغط على مفتاح الإدخال يتم تخزين هذا الإسم فى متغير الذاكرة (mname).

٥ - الأمـــر (APPEND)

يستخدم هذا الأمر فى إضافة سجلات جديدة إلى ملف قاعدة البيانات. وهو يستخدم شاشة الإدخال المستخدمة سواء كانت شاشة الإدخال المبدئية (Default) أو شاشة الإدخال التى يتم تصميمها من خلال البرنامج.

والصورة العامة للأمر كالآتى :

APPEND [BLANK]

ويستخدم الإختيار (BLANK) لإضافة سجل خال فى نهاية ملف قاعدة البيانات حتى يمكن إحلال بيانات الحقول مكان الحقول الخالية من خلال البرنامج.

مـثال

يمكن كتابة الأوامر التالية :

```
.USE Cadets
.APPEND
```

وعند الضغط على مفتاح الإدخال تظهر الشاشة كما فى المثال بشكل (٣٨ - ١).

NAME	[ ]	TELEPHONE	[ ]
ADDRESS	[ ]	CLASS	[ ]
NATIONALITY	[ ]	HOBBIES	[ ]
FATHER NAME	[ ]	BIRTH DATE	[ ]
FATHER JOB	[ ]	RELIGION	[ ]
MOTHER NAME	[ ]	NOTES	[ ]

شكل (٣٨ - ١)

وعند الإنتهاء من إدخال بيانات الحقول يتم الضغط على مفتاح الإدخال فيتم تخزين هذا السجل. كما يمكن الضغط على مفتاحى (Ctrl-End) لتخزين جميع السجلات التى تمت إضافتها والعودة إلى مشيرة النقطة (Dot Prompt).

٦ - الأمـــر (APPEND FROM)

يستخدم هذا الأمر فى نسخ سجلات من ملف إلى نهاية ملف قاعدة البيانات المفتوح. ولايشترط أن يكون الملف المنسوخ منه ملف قاعدة بيانات كما سنلاحظ من الصور المختلفة للأمر وإنما يمكن أن يكون ملف جدول إلكترونى (Spread Sheet) أو أى ملف آخر مكتوب بشفرة الآسكى (ASCII Code). وهناك صورتان للأمر، الصورة الأولى تستخدم للنسخ من ملف قاعدة بيانات آخر، والصورة الثانية تستخدم للنسخ من ملفات أخرى.

أ - الصورة الأولى

APPEND FROM <Filename> [FOR <condition>]

حيث (filename) هو إسم الملف المنسوخ منه ولايكتب فيه الإمتداد (Extension) حيث أن البرنامج يضيف الإمتداد (.dbf) آليا. و (condition) هو الشرط الذى يحدد السجلات المطلوب نقلها ويجب ملاحظة أن الحقول المشتركة فى الملفين فقط هى التى يتم نسخ بياناتها. ولايشترط أن تكون بنفس الترتيب فى الملفين. وإذا كان أى حقل فى الملف المنسوخ منه أكبر من الحقل المقابل فى الملف الآخر يتم حذف الحروف الزائدة (Truncation) إذا كان الحقل حرفيا كما يتم إستبدال الأرقام بنجوم (Astriks) إذا كان الحقل عدديا.

مثال

نفرض أنه قد تم إنشاء ملف قاعدة بيانات إسمه (School1) ويراد إضافة بيانات الطلبة الناجحين فقط فى المدرسة إلى ملف آخر إسمه (Cadets) لتنفيذ ذلك يتم كتابة الأوامر التالية :

```
USE Cadets
APPEND FROM School1 FOR grade>50
```

فى هذه الحالة يتم إضافة السجلات التى تحقق الشرط فقط إلى الملف (Cadets).

ب - الصورة الثانية

APPEND FROM <Filename> TYPE <Filetype>

وتستخدم هذه الصورة فى إضافة سجلات من ملفات ليست مكتوبة بواسطة (DBase III+) وهى تشمل الأنواع التالية :

١ - الملفات ذات الإمتداد (SDF) وهى الملفات المكتوبة بشفرة الآسكى (ASCII Code). ويتم نسخها حرفا حرفا بحيث ينتهى كل سجل بالكود

الخاص بالإدخال (Carriage Return).

٢ - الملفات ذات الإمتداد (SYLK) وهى الملفات المكتوبة بواسطة برنامج (VisiCalc) وهو برنامج جداول إلكترونية. حيث يتم تحويل السطور (rows) إلى سجلات والأعمدة (Columns) إلى حقول.

٣ - الملفات ذات الإمتداد (WKS) وهى الملفات المكتوبة بواسطة برنامج لوتس ١٢٣ (Lotus 123) حيث يتم تحويل السطور إلى سجلات والأعمدة إلى حقول.

ويجب ملاحظة كتابة الإمتداد عند كتابة إسم الملف المنسوخ منه (Filename) ، وكذلك كتابة نوع الملف المنسوخ منه بعد كلمة (TYPE) وذلك كالآتى مثلا :

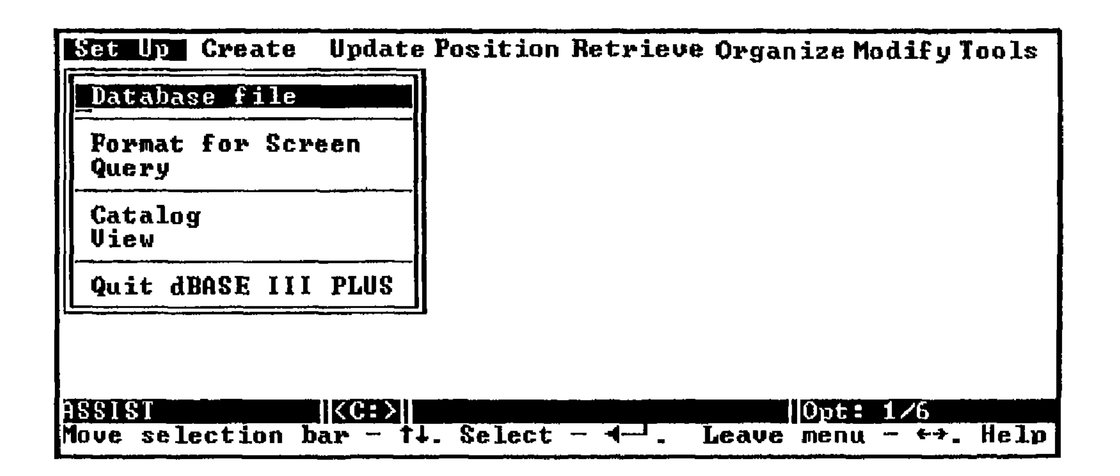
```
APPEND FROM ACCOUNTS.WKS TYPE WKS
```

٧ - الأمـــر (ASSIST)

ويستخدم هذا الأمر فى تشغيل القوائم الرئيسية لبرنامج المساعد (ASSISTANT) التى يمكن عن طريقها إنشاء ملف قاعدة البيانات والملفات المرتبطة به مثل ملفات الفهرس (Index Files) وملفات التشكيل (Format Files) وملفات البحث (Query Files) وملفات المنظر(View Files) بالإضافة إلى ملفات التقارير(Reports) والعناوين المختصرة (Labels). وتسمح هذه القوائم كذلك بتعديل هذه الملفات وإجراء عمليات التصحيح والبحث والعرض لأى بيانات مطلوبة.

ويتم تنفيذ أى اختيار من القوائم عن طريق تحريك مؤشر القائمة حتى يصل إلى الإختيار المطلوب ثم الضغط على مفتاح الإدخال. ويلاحظ عند الوصول إلى الإختيار النهائى المطلوب تنفيذه ظهور الأمر المقابل له بالكامل على السطر أعلى عمود الحالة (Status Bar) والذى يسمى سطر الفعل (Action Line). وهذا الأمر الذى يظهر هو نفس الأمر الذى يمكن للمستخدم كتابته من مشيرة النقطة (Dot Prompt) للحصول على نفس النتيجة. وتتيح هذه الطريقة للمستخدم التعرف على الشكل (Syntax) الخاص بأى أمر أثناء العمل من خلال قوائم برنامج المساعد (Assistant). كما يمكن الحصول على أى معلومات إضافية على الشاشة وذلك عن طريق عرض شاشة مساعدة (Help) توضح للمستخدم شكل الأمر (Syntax) بالإضافة إلى شرح خصائص استخدام هذا الأمر.

ويمكن تشغيل قوائم برنامج المساعد (Assistant) عن طريق كتابة الأمر (ASSIST) من مشيرة النقطة (Dot Prompt). كما يمكن تشغيلها أيضا عن طريق الضغط على مفتاح (F2) ويلاحظ فى هذه الحالة ظهور القائمة الرئيسية على الشاشة. انظر الشكل (٣٨ - ٢).



شكل (٣٨ - ٢)

وهذه القائمة تحتوى على ثمانية إختيارات يمكن تلخيصها كالآتى :

أ - قائمـــة التجهيـــز (Set Up)

تستخدم هذه القائمة فى فتح ملف قاعدة البيانات والملفات المرتبطة به كما تستخدم أيضا فى الخروج من البرنامج. وعند فتح أى ملف تظهر قائمة بوحدات الأقراص المتاحة لاختيار وحدة الأقراص التى تحتوى على الملف المطلوب فتحه. وعند فتح ملف قاعدة البيانات وبعد اختيار وحدة الأقراص والملف المطلوب يظهر سؤال عما إذا كان الملف مفهرسا (Indexed) أم لا. فإذا كان الملف مفهرسا يتم كتابة (Y) فتظهر قائمة بأسماء ملفات الفهرس ويتم اختيار ملفات الفهرس الخاصة بملف قاعدة البيانات المفتوح على ألا يزيد عدد ملفات الفهرس المفتوحة عن سبعة ملفات. وكل ملف يتم اختياره تظهر أمامه علامة ( ) مع ملاحظة أن أول ملف يتم اختياره يصبح هو ملف الفهرس الرئيسى (Master) بصرف النظر عن ترتيب هذا الملف فى القائمة. ويتم تخزين هذه الإختيارات عن طريق الضغط على مفتاح السهم شمال (-->).

وعند فتح ملف التشكيل (Format) يتم تحديد وحدة الأقراص فتظهر قائمة بملفات التشكيل الموجودة. ويتم اختيار ملف التشكيل المطلوب استخدامه. ويمكن فتح ملف البحث (Query) وذلك باختيار ملف البحث المطلوب من قائمة الملفات التى تظهر على الشاشة.

كما يمكن فتح الكتالوج إذا كان قد سبق إنشاء ملف كتالوج حيث يتم اختيار ملف الكتالوج المطلوب من قائمة الملفات التى تظهر على الشاشة. وفى هذه الحالة لاتظهر على الشاشة دائما إلا الملفات المخزنة فى هذا الملف. فعند فتح ملف قاعدة البيانات أو الملفات المرتبطة به ، لاتظهر إلا الملفات الموجودة فى ملف الكتالوج المفتوح. فى حين لو لم يتم فتح ملف الكتالوج تظهر جميع الملفات الموجودة على وحدة الأقراص المستخدمة (Current Drive).

ويمكن فتح ملف المنظر (View File) بنفس الطريقة مثل الملفات السابقة. حيث يتم تحديد وحدة الأقراص المستخدمة ثم اختيار ملف المنظر من قائمة ملفات المنظر التى تظهر على الشاشة. وللخروج من البرنامج يتم اختيار (+Quit DBase III) ويؤدى هذا إلى إغلاق جميع الملفات المفتوحة والعودة إلى نظام التشغيل.

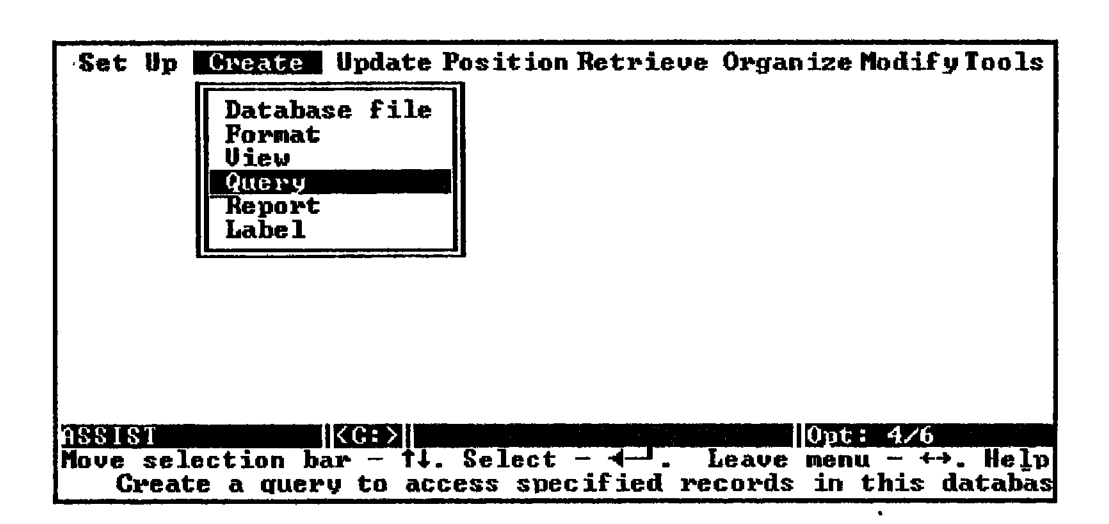
ب - قائمـــة الإنشـــاء (Create)

تستخدم هذه القائمة فى إنشاء الملفات ولكل نوع من الملفات يتم اختيار وحدة الأقراص المطلوب تخزين الملف فيها ثم يتم كتابة إسم الملف. وإذا كان قد سبق فتح كتالوج فإن هذا الملف الذى يتم إنشاؤه يدخل فى الكتالوج. انظر شكل (٣٨ - ٣).

والإختيار (Database File) يستخدم فى إنشاء ملف قاعدة بيانات. ارجع إلى الأمر (CREATE).

والإختيار (Format) يستخدم فى إنشاء ملف تشكيل (Format File) وهو ملف يؤدى إلى التحكم فى شكل شاشة الإدخال التى يتم عن طريقها إدخال البيانات. إرجع إلى الأمر (CREATE SCREEN).

والإختيار (View) يستخدم فى إنشاء ملف المنظر (View File) وهو يسمح بالتعامل مع عدة ملفات قواعد بيانات فى نفس الوقت. إرجع إلى الأمر (CREATE VIEW).



شكل (٣٨ - ٣)

والإختيار (Query) يستخدم فى إنشاء ملف بحث (Query File) وهذا يساعد على ترشيح ملف قاعدة البيانات (Filtering) للحصول على البيانات المطلوبة. إرجع إلى الأمر (CREATE QUERY).

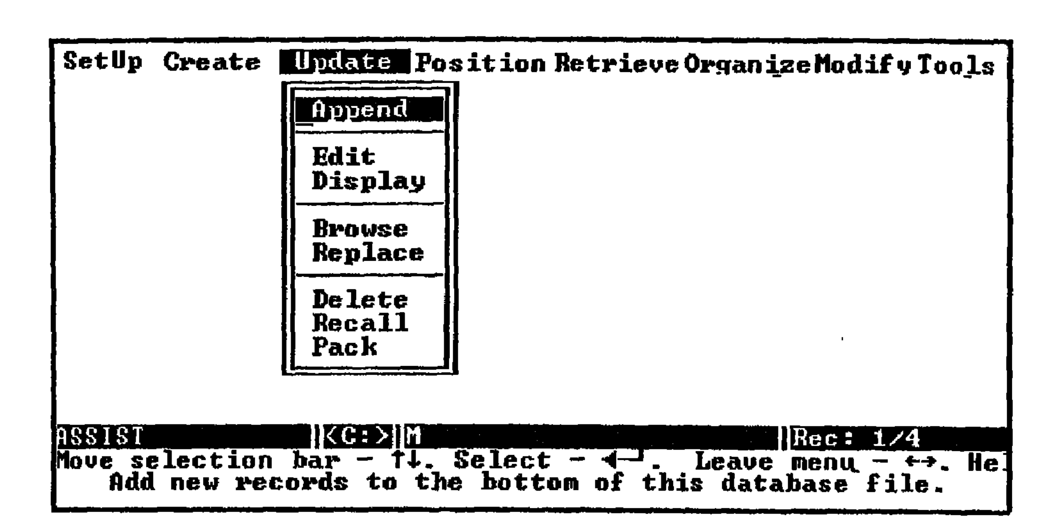
والإختيار (Report) يستخدم فى إنشاء ملف التقرير الذى يحتوى على الإمتداد (.Frm). إرجع إلى الأمر (CREATE REPORT).

والإختيار (Label) يستخدم فى إنشاء ملف العناوين المختصرة (Label). إرجع إلى الأمر (CREATE LABEL).

جـ - قائمة التحديث (Update)

وتستخدم هذه القائمة فى تحديث البيانات المخزنة فى قاعدة البيانات فيمكن من خلالها إضافة سجلات أو مسح سجلات أو تصحيح بيانات سجلات معينة فى الملف. وعند فتح ملفات الفهرس (Index Files) فإن هذه الملفات يتم تحديثها آليا مع تحديث بيانات قاعدة البيانات. انظر شكل (٣٨ - ٤)

والإختيار (Append) يستخدم فى إضافة سجلات جديدة فى نهاية الملف. إرجع إلى الأمر (APPEND).



شكل (٣٨ - ٤)

والإختيار (Edit) يستخدم فى تعديل بيانات الملف سجلا سجلا ، وذلك حسب السجل الذى يقف عنده مؤشر الملف. إرجع إلى الأمر (EDIT).

والإختيار (Display) يستخدم فى عرض بيانات سجل معين. إرجع إلى الأمر (DISPLAY).

والإختيار (Browse) يستخدم فى تعديل بيانات الملف ، بالإضافة إلى عرض حتى ١٧ سجلا على الشاشة. إرجع إلى الأمر (BROWSE).

والإختيار (Replace) يسمح باجراء تعديلات مجمعة (Batch) للملف قاعدة البيانات. ويمكن استخدامه مثلا فى تعديل مرتب جميع الموظفين فى شركة معينة عن طريق ضرب حقل المرتب (Salary) فى نسبة ثابتة مثل (5 %).

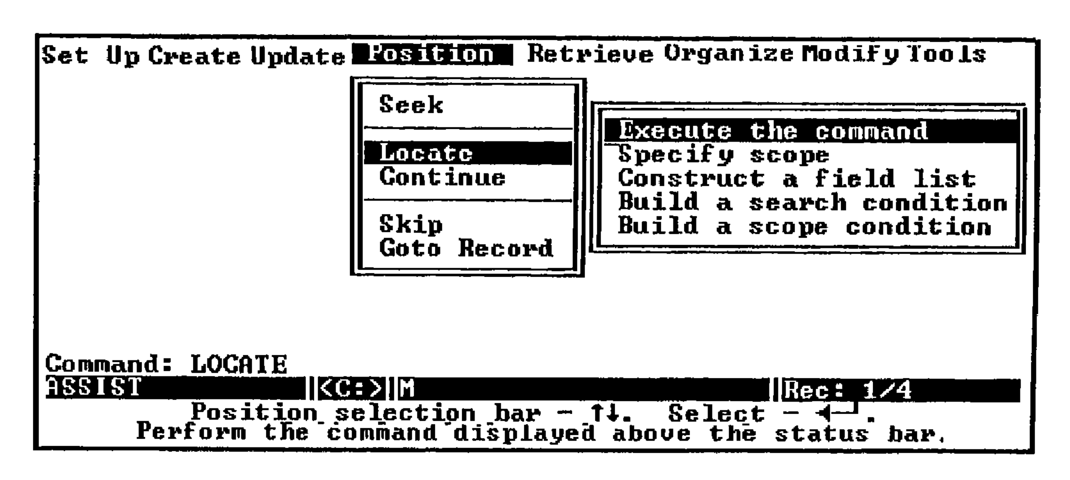
والإختيار (Delete) يستخدم فى وضع علامات (Marks) على السجلات المطلوب مسحها تمهيدا لمسحها تماما بواسطة الاختيار (Pack).

والإختيار (Recall) يستخدم فى إستعادة السجلات التى تم وضع علامات عليها لمسحها. ويمكن فى هذه الحالة إستعادة بعض السجلات أو كلها حسب الحاجة.

والإختيار (Pack) يستخدم فى المسح النهائى للسجلات التى تم وضع علامات عليها لمسحها.

د - قائمة المكان (Position)

وتستخدم هذه القائمة فى توجيه مؤشر السجلات (Record Pointer) إلى سجل محدد وذلك حتى يمكن تحديث هذا السجل أو عرض بياناته. وفى كل اختيار من اختيارات هذه القائمة يقوم البرنامج بعرض القوائم الفرعية التى يتم عن طريقها إدخال الشرط أو الشروط التى يتم بناء عليها توجيه المؤشر إلى سجل محدد. انظر شكل (٣٨ - ٥)



شكل (٣٨ - ٥)

والإختيار (Seek) يستخدم فقط عندما يكون الملف مفهرسا (Indexed) وهو يوجه مؤشر السجلات إلى أول سجل يحقق الشرط الذى يتم إدخاله. إرجع إلى الأمر (SEEK).

والإختيار (Locate) يستخدم فى توجيه مؤشر السجلات (Record Pointer) إلى أول سجل يحقق الشرط الذى يتم إدخاله. إرجع إلى الأمر (LOCATE).

والإختيار (Continue) يستخدم فقط بعد استخدام الاختيار (Locate) وذلك لعرض السجل الثانى الذى يحقق الشرط الذى سبق إدخاله.

والإختيار (Skip) يستخدم فى تحريك مؤشر السجلات (Record Pointer) إلى سجلات تالية أو سجلات سابقة حسب العدد الذى يتم إدخاله. إرجع إلى الأمر (SKIP).

والإختيار (Goto Record) يستخدم فى تحديد رقم سجل معين يراد وضع المؤشر عنده. إرجع إلى الأمر (GOTO)

هـ - قائمة الإسترجاع (Retrieve)

وتستخدم هذه القائمة فى عرض بيانات سجلات معينة على الشاشة أو طباعتها على الطابعة. انظر الشكل (٣٨ - ٦)

```
SetUp Create Update Position Retrieve Organize Modify Tools
                             List
                             Display
                             Report
                             Label
                             Sum
                             Average
                             Count
ASSIST          |<C:>|M                    |Rec: 1/4
Move selection bar - ↑↓. Select - ◄┘. Leave menu - ↔. He
```

شكل (٣٨ - ٦)

والإختيار (List) يستخدم لعرض بيانات جميع السجلات المطلوبة على الشاشة أو طباعتها على الطابعة حسب الحاجة. إرجع إلى الأمر (LIST).

والإختيار (Display) يستخدم لنفس الهدف مثل الاختيار (List) ولكن هناك بعض الاختلافات بينهما. إرجع إلى الأمر (DISPLAY).

والإختيار (Report) يستخدم فى عرض البيانات على الشاشة أو طباعتها

على الطابعة حسب شكل التقرير (Report) الذى سبق إنشاؤه. إرجع إلى الأمر (REPORT).

والإختيار (Label) يستخدم فى عرض البيانات على الشاشة أو طباعتها على الطابعة حسب شكل التقرير المختصر (Label) الذى سبق إنشاؤه. إرجع إلى الأمر (LABEL).

والإختيار (Sum) يستخدم لتجميع الحقول العددية لمجموعة من السجلات التى يتم اختيارها. إرجع إلى الأمر (SUM).

والإختيار (Average) يستخدم لحساب القيم المتوسطة لكل الحقول العددية فى مجموعة من السجلات التى يتم اختيارها. إرجع إلى الأمر (AVERAGE).

والإختيار (Count) يستخدم فى حساب عدد السجلات التى تحقق شرطا أو شروطا معينة. إرجع إلى الأمر (COUNT).

و - قائمة التنظيم (Organize)

تستخدم هذه القائمة فى تنظيم السجلات وترتيبها داخل ملف قاعدة البيانات. وكل اختيار من اختيارات هذه القائمة يؤدى إلى إنشاء ملف جديد. انظر شكل (٣٨ - ٧)

```
SetUp Create Update Position  Retrieve Organize  Modify Tools
                                       Index
                                       Sort
                                       Copy

        The index key can be any character, numeric, or
        date expression involving one or more fields in
        the database file.  It is usually a single field.
        Enter an index key expression:

ASSIST              ||<C:>||M                     ||Rec: 1/4
Move selection bar - ↑↓. Select - ◄─┘.  Leave menu - ←→. He
```

شكل (٣٨ - ٧)

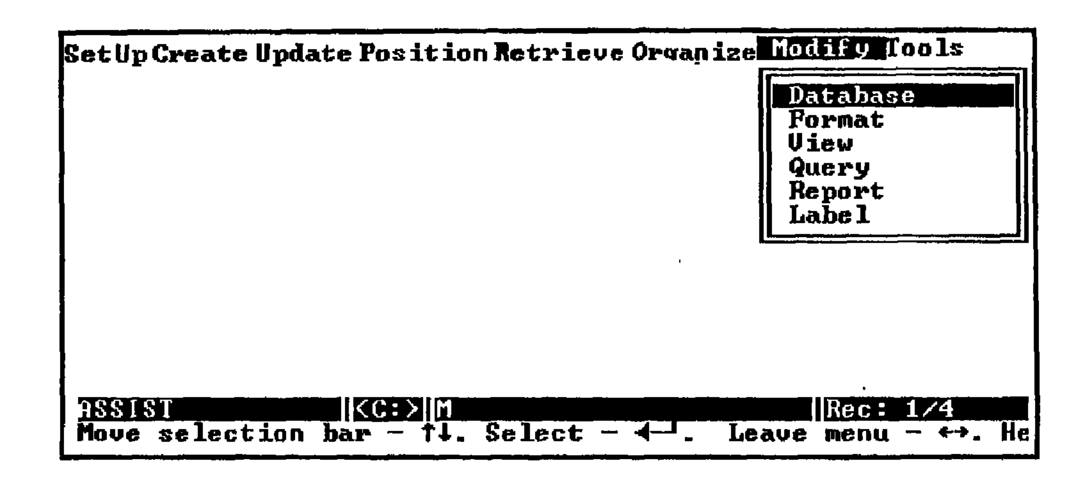
والإختيار (Index) يؤدى إلى إنشاء ملف فهرسى (Index File). وعند فتح هذا الملف من خلال قائمة التجهيز (Setup) فإن سجلات ملف قاعدة البيانات تظهر دائما مرتبة حسب الحقل الفهرسى (Key Field) الذى تم اختياره وذلك رغم عدم حدوث أى تغيير فى الأماكن الفعلية للسجلات. إرجع إلى الأمر (INDEX).

والإختيار (Sort) يؤدى إلى إنشاء ملف قاعدة بيانات جديد مرتب بالترتيب المطلوب. إرجع إلى الأمر (SORT).

والإختيار (Copy) يستخدم فى عمل نسخة من ملف قاعدة البيانات. ويمكن نسخ الملف كله أو نسخ مجموعة من السجلات التى تحقق شرطا أو شروطا معينة فقط. كما يمكن أيضا نسخ بيانات مجموعة من الحقول فقط وليس كل الحقول. إرجع إلى الأمر (COPY).

ز - قائمة التعديل (Modify)

أنظر الشكل (٣٨ - ٨)



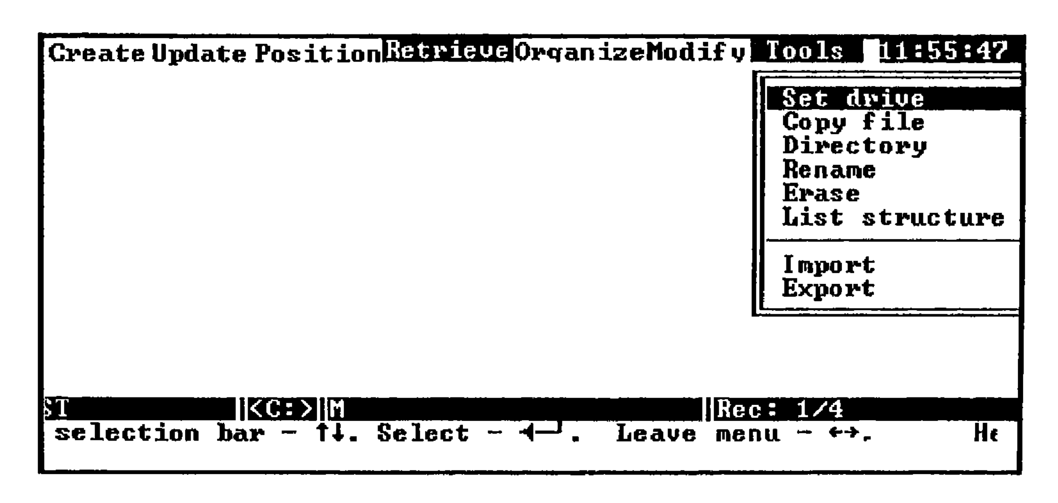
شكل (٣٨ - ٨)

وهذه القائمة تستخدم فى تعديل الملفات التى سبق إنشاؤها من خلال قائمة الإنشاء (Create). وهى تحتوى على نفس الاختيارات الموجودة فى قائمة الإنشاء

(Create) ما عدا الكتالوج. وعند اختيار نوع الملف المطلوب تعديله سواء كان ملف قاعدة البيانات أو الملفات المرتبطة به فإن البرنامج يعرض أسماء الملفات الموجودة على وحدة الأقراص من نفس النوع. إرجع إلى الأمر (MODIFY).

حـ - قائمة الأدوات (Tools)

وهذه القائمة تؤدى بعض الوظائف التى يقوم بها نظام التشغيل مثل عرض فهرس الملفات (Directory) ونسخ الملف (Copy) وتغيير إسم ملف (Rename) وهكذا. انظر الشكل (٣٨ - ٩)



شكل (٣٨ - ٩)

والإختيار (Set Drive) يستخدم فى تحديد وحدة الأقراص المستخدمة وذلك بإدخال رمز وحدة الأقراص (A, B, C, ...) التى تحتوى على الملفات المطلوب استخدامها.

والإختيار (Copy File) يستخدم فى نسخ أى نوع من الملفات. وهو لذلك يختلف عن الاختيار (Copy) فى قائمة التنظيم (Organize) حيث أن الاختيار (Copy) يستخدم فى نسخ ملفات قاعدة البيانات فقط أى التى تحتوى على الإمتداد (dbf.). إرجع إلى الأمر (COPY FILE).

والإختيار (Directory) يستخدم فى عرض قائمة أسماء الملفات الموجودة فى وحدة الأقراص المستخدمة. إرجع إلى الأمر (DIR).

والإختيار (Rename) يستخدم لتغيير إسم أى ملف بشرط ألا يكون مفتوحا فى هذا الوقت. إرجع إلى الأمر (RENAME).

والإختيار (Erase) يستخدم فى مسح أى ملف من القرص بشرط ألا يكون مفتوحا فى هذا الوقت. إرجع إلى الأمر (ERASE).

والإختيار (List Structure) يستخدم فى عرض تركيب الملف المفتوح. وهذا الاختيار يسمح أيضا بطباعة هذا التركيب (Structure) على الطابعة. إرجع إلى الأمر (LIST STRUCTURE).

والإختيار (Import) يستخدم فى تحويل الملفات التى تمت كتابتها عن طريق برامج أخرى غير برنامج (+DBase III) إلى ملفات يمكن استخدامها مع برنامج (+DBase III). إرجع إلى الأمر (IMPORT).

والإختيار (Export) يستخدم فى تحويل الملفات المكتوبة بواسطة برنامج (+DBase III) إلى ملفات يمكن استخدامها بواسطة برامج أخرى. إرجع إلى الأمر (EXPORT).

٨ - الأمـــــر (AVERAGE)

ويستخدم هذا الأمر فى حساب القيم المتوسطة للحقول العددية لمجموعة من السجلات. والصورة العامة للأمر كالآتى :

```
AVERAGE[<expression list>][<scope>]
[WHILE <condition>]
[FOR <condition>] [TO <memvar list>]
```

ويلاحظ هنا أن جميع الإختيارات إختيارية أى يمكن كتابتها أو عدم كتابتها. وعند عدم كتابة أى شىء بعد الأمر (AVERAGE) فإن ذلك يؤدى إلى حساب القيم المتوسطة لجميع الحقول العددية لجميع سجلات قاعدة البيانات المفتوحة وعرض هذه المتوسطات على الشاشة.

أما الإختيارات الموجودة فإنها تحدد الحقول المطلوب حساب متوسطها كما تحدد المدى (Scope) أى السجلات المطلوب البحث خلالها كما تحدد الشروط المطلوب البحث بناء عليها ثم تحدد أسماء متغيرات الذاكرة التى يتم تخزين القيم المتوسطة فيها.

والإختيار الأول (expression list) يتم من خلاله كتابة أسماء الحقول المطلوب حساب متوسطها.

والإختيار الثانى (scope) يتم من خلاله تحديد مدى السجلات المطلوب البحث خلاله. حيث يمكن البحث خلال كل الملف (ALL) أو خلال السجلات التالية للسجل الذى يقف عنده المؤشر (Rest) و .... وهكذا.

والإختيار الثالث (WHILE <condition>) يؤدى إلى البحث عن السجلات التى تحقق شرطا أو شروطا معينة.

والإختيار الرابع (FOR <condition>) يؤدى نفس العمل ، أى يبحث عن السجلات التى تحقق شرطا أو شروطا معينة. ولكن الاختيار (WHILE) أسرع فى الوصول الى السجلات التى تحقق الشروط.

والإختيار الخامس (TO <memvar list>) يؤدى إلى إنشاء متغيرات الذاكرة (memvar list) التى يتم فيها تخزين المتوسطات المحسوبة وذلك بنفس الترتيب الذى يتم به كتابة هذه المتغيرات.

مثال

للحصول على متوسطات حقل ساعات العمل (w_hours) وحقل المرتب (salary) لكل ملف قاعدة البيانات يتم كتابة السطر التالى :

```
AVERAGE w_hours, salary TO avg_hr, avg-sal
```

فى هذه الحالة يحتوى المتغير (avg_hr) على متوسط ساعات العمل كما يحتوى المتغير (avg_sal) على متوسط المرتبات.

٩ - الأمــــر (BROWSE)

ويؤدى هذا الأمر إلى عرض شاشة لتعديل بيانات السجلات وإدخال سجلات جديدة. وهو يؤدى نفس العمل الذى يتم عند الدخول فى قائمة التحديث (Update) واختيار (Browse). ولكن كتابته من مشيرة النقطة (Dot Prompt) أو من خلال البرنامج تعطى مرونة أكبر فى تحديد الحقول المطلوب عرضها على الشاشة.

والصورة العامة لهذا الأمر كالآتى :

BROWSE [FIELDS <field list>][LOCK <N>]
[FREEZE] <field>][NOFOLLOW][NOMENU][WIDTH<N>]
[NOAPPEND]

ويلاحظ هنا أن جميع الإختيارات اختيارية أى يمكن كتابة الأمر (BROWSE) دون كتابة أى شىء بعده. وفى هذه الحالة تظهر شاشة تعديل البيانات وتحتوى على بيانات ١٧ سجلا. ويمكن عرض عدد أكبر من السجلات عن طريق الضغط على مفتاح (F1) لإخفاء قائمة المساعدة (Help) الموجودة أعلى الشاشة وتظهر الحقول على هيئة أعمدة. وعند الضغط على مفتاح (F10) يظهر عمود القوائم (menu bar) الخاص بالأمر (Browse). وقد سبق شرح هذه القوائم فى الفصل الخاص بتعديل البيانات. ويمكن إدخال معظم هذه الإختيارات مع الأمر (BROWSE) من خلال الاختيارات الموجودة مع الأمر والتى يمكن تلخيصها كالآتى : انظر الشكل (٣٨ - ١٠).

الإختيار (<FIELDS <field list) ويستخدم لتحديد أسماء الحقول المطلوب عرض البيانات الخاصة بها. ويمكن من خلال هذا الإختيار تحديد أى ترتيب يراد عرض الحقول به. وذلك عن طريق كتابة أسماء الحقول (Field List) بنفس هذا الترتيب.

والإختيار (<LOCK <N) يستخدم لتثبيت عدد من الحقول المتجاورة فى يسار الشاشة والتى لاتتحرك عند الضغط على مفتاحى (<---,Ctrl) أو (<---,Ctrl) لعمل زحزحة أفقية (Horizontal Scrolling) ويؤدى هذا إلى عرض الحقول المختفية من الملف أمام مجموعة من الحقول تكون ظاهرة دائما على الشاشة. ويتم تحديد عدد الحقول المراد تثبيتها عن طريق العدد N.

```
CURSOR   <-- -->     |        UP    DOWN  |   DELETE      | Insert Mode:   Ins
 Char:      ←   →    | Record:  ↑     ↓   | Char:    Del  | Exit:         ^End
 Field: Home End     | Page:   PgUp  PgDn | Field:   ^Y   | Abort:         Esc
 Pan:      ^←  ^→    | Help:    F1        | Record:  ^U   | Set Options: ^Home

NAME------------------ ADDRESS----------- PHONE------ FATHER---------------
MOHAMED AHMED FATHY    12-ain shams       56526756    AHMED
AHMED HASAN            40-ahran-street    6789889     HASAN
WALAA MOSTAFA          CITY               6394588     MOSTAFA
HAYTHAM MOSTAFA        NASR CITY          7428953     MOSTAFA
FATEN KAMAL            NASR CITY          4676787     KAMA HEREK

BROWSE          |<C:>|N                    |Rec: 1/5          |        |

                          View and edit fields.
```

شكل (٣٨ - ١٠)

والإختيار (FREEZE <field>) يستخدم لتجميد حقل معين حتى يصبح هو الحقل الوحيد المسموح بتعديله. وبالرغم من عرض باقى الحقول على الشاشة إلا أن المستخدم لايستطيع تعديل أى حقل آخر غير هذا الحقل.

والإختيار (NOFOLLOW) يستخدم فقط مع الملفات المفهرسة (Indexed). وهو يؤدى إلى إنتقال المؤشر إلى السجل الجديد فى حالة تغيير الحقل الفهرسى(Key Field). وذلك لأن تغيير الحقل الفهرسى يؤدى إلى تغيير ترتيب السجلات فى حين يظل المؤشر مكانه.

والإختيار (NOMENU) يستخدم لمنع استخدام عمود الاختيارات (Menu Bar) الخاص بالأمر (BROWSE).

والإختيار (WIDTH <N>) يستخدم لتحديد عرض الحقل المراد ظهوره بالحروف حيث يمثل العدد N عدد الحروف فى الحقل.

والإختيار (NOAPPEND) يستخدم لمنع المستخدم من إضافة أى سجلات جديدة إلى الملف.

١٠ - الأمــــر (CALL)

ويتيـح هـذا الأمـر لمخطـط البـرامـج كتـابـة بـرامـج منفصلـة بلغـة التجميـع (Assembly Language) وتحميلها داخل ذاكرة الحاسب وهذه البرامج يجب تحويلها أولا إلى الشفـرة الثنـائيـة (Binary Code). ويتـم تحميلهـا فـى الـذاكـرة بـاستخـدام الأمـر (LOAD) كما يمكن تنفيذها داخل البرنامج المكتوب بواسطة برنامج (+DBase III) باستخدام الأمر (CALL) حيث يتم كتابة هذا الأمر وبعده إسم البرنامج المطلوب تنفيذه. ويتم تحويل البرنامج من برنامج منفذ (Executable) إلى برنامج ثنائى (Binary) عن طريق الأمر (EXE2BIN) وهو أمر من أوامر نظام التشغيل (MS-DOS).

١١ - الأمــــر (CANCEL)

ويستخدم هذا الأمر لإيقاف تنفيذ البرنامج والعودة إلى مشيرة النقطة والصورة العامة له كالآتى :

```
CANCEL
```

وهو لايحتاج إلى معاملات أخرى.

مـثال

لإيقاف تنفيذ البرنامج عند ضغط المستخدم على الحرف X يتم كتابة الأوامر التالية :

```
IF  Choice = 'X'
    CANCEL
ENDIF
```

١٢ - الأمــــر (CHANGE)

ويـؤدى هـذا الأمـر إلى عـرض شـاشـة تصحيـح مثـل الشـاشـة التـى تظهـر مـع استخدام الإختيار (Edit) من قائمة التحديث (Update) ولكن كتابة الأمر من مشيرة النقطة (Dot Prompt) أو من خلال البرنامج تتيح التحكم فى الحقول المطلوب تعديلها وكذلك تحديد السجل المطلوب تعديله بسهولة والصورة العامة للأمر كالآتى :

CHANGE [<scope>] [FIELDS <field list>]
[WHILE <condition>] [FOR <condition>]

ويلاحظ أن جميع الاختيارات إختيارية أى يمكن كتابة الأمر (CHANGE) دون كتابة أى شىء بعده. ويؤدى ذلك إلى ظهور شاشة التصحيح التى يتم من خلالها تعديل السجلات على التوالى سجلا تلو الآخر بدءا من السجل الذى يقف عنده مؤشر السجلات (Record Pointer). ويتم الإنتقال من السجل إلى السجل التالى بالضغط على مفتاح (PgDn).

أما الإختيارات الموجودة فإنها تحدد الحقول المطلوب تعديلها وكذلك السجل المطلوب تعديله. ويمكن تلخيص هذه الإختيارات كالآتى :

الإختيار (<scope>) ويستخدم فى تحديد مدى السجلات المطلوب البحث خلالها. وقد يكون البحث خلال كل الملف (ALL) أو بدءا من السجل الذى يقف عند المؤشر وحتى آخر الملف (Rest) كما يمكن تحديد سجل معين بكتابة رقمه.

والإختيار (<FIELDS <field list) يستخدم لتحديد أسماء الحقول المطلوب تعديلها وهى الحقول التى تظهر على شاشة الإدخال.

والإختيار (WHILE <condition>) يستخدم للبحث عن السجلات التى تحقق الشرط أو الشروط التى يتم إدخالها كما يستخدم الاختيار (<FOR <condition) لنفس الغرض.

ملاحظة

الأمر (CHANGE) والأمر (EDIT) متماثلان تماما.

مثال

لتعديل حقول الإسم والعنوان وتاريخ الميلاد فى ملف بيانات الطلبة (Cadets) يمكن كتابة السطور التالية :

```
USE Cadets
CHANGE FIELDS name, address, birth_d
```

١٣ - الأمـــر (CLEAR)

ويستخدم هذا الأمر فى مسح الشاشة ووضع مؤشر الشاشة عند أعلى نقطة من اليسار وهى النقطة التى إحداثياتها (صفر ، صفر) كما يمكن مسح جزء فقط من الشاشة عن طريق كتابة الأمر (CLEAR....@) مع كتابة الإحداثيات المطلوب المسح بدءا منها بعد الحرف @. وذلك كالآتى مثلا :

```
@ 10,20 CLEAR
```

١٤ - الأمـــر (CLEAR ALL)

ويؤدى هذا الأمر إلى إغلاق جميع ملفات قواعد البيانات المفتوحة والملفات المرتبطة بها مثل ملفات الفهرس (Index Files) وملفات التشكيل (Format Files) و .... الخ. كما يؤدى هذا الأمر أيضا إلى مسح متغيرات الذاكرة (Memory Variables).

١٥ - الأمـــر (CLEAR FIELDS)

ويستخدم هذا الأمر فى مسح الحقول التى سبق تحديدها بواسطة الأمر (SET FIELDS TO). والأمر (SET FIELDS TO) يؤدى إلى تحديد الحقول التى يتم استخدامها فقط من الملف فى حين تبقى باقى الحقول مغلقة وغير مستخدمة. ولذلك يستخدم الأمر (CLEAR FIELDS) فى إعادة الملف إلى وضعه المبدئى حيث تصبح جميع الحقول عاملة (Active).

١٦ - الأمـــر (CLEAR GETS)

يستخدم هذا الأمر فى مسح كل المتغيرات التى تم إنشاؤها بواسطة الأمر (GET...@). وهذه المتغيرات لاتزيد عن ١٢٨ متغيرا وذلك حسب الوضع المبدئى (Default) للبرنامج. ويفضل دائما عند زيادة هذه المتغيرات أن يتم مسح بعضها بواسطة هذا الأمر وذلك بعد تخزينها فى متغيرات ذاكرة خصوصا فى الحالات التى يراد فيها إنشاء شاشات إدخال من عدة صفحات.

١٧ - الأمـــر (CLEAR MEMORY)

يستخدم هذا الأمر فى مسح متغيرات الذاكرة من الذاكرة المؤقتة للحاسب. وهو يماثل الأمر (RELEASE ALL) ولكن الإختلاف بين الأمرين أن الأمر (CLEAR MEMORY) يمسح كل متغيرات الذاكرة سواء كانت عامة (Public) أو خاصة (Private) أما الأمر (RELEASE ALL) فإنه يمسح المتغيرات الخاصة فقط.

١٨ - الأمـــر (CLOSE)

ويستخدم هذا الأمر لإغلاق الملفات المفتوحة وهو يكون على إحدى صورتين :

```
CLOSE < file type >
CLOSE ALL
```
أو

والصورة الأولى يتم عن طريقها إغلاق نوع معين من الملفات أما الصورة الثانية فإنها تستخدم فى إغلاق جميع الملفات المفتوحة. ولتوضيح ذلك يمكن كتابة الأوامر التالية :

```
CLOSE DATABASES
CLOSE INDEX
CLOSE FORMAT
CLOSE PROCEDURE
CLOSE ALL
```

١٩ - الأمـــر (CONTINUE)

ويستخدم هذا الأمر مع الأمر (LOCATE) لتوجيه مؤشر السجلات (Record Pointer) إلى السجل المطلوب حيث يستخدم الأمر (CONTINUE) فى تحريك المؤشر إلى السجل الثانى الذى يحقق الشرط ثم السجل الذى يليه وهكذا. إرجع إلى الأمر (LOCATE).

٢٠ - الأمـــر (COPY)

وهو يشبه الأمر (APPEND FROM) ولكنه لايضيف سجلات فى نهاية الملف المفتوح بل ينسخ الملف المفتوح كله أو جزءا منه فى ملف آخر جديد. وهناك صورتان لهذا

الأمر. الصورة الأولى تستخدم فى نسخ ملف قاعدة البيانات كله أو جزء منه فى ملف قاعدة بيانات آخر. والصورة الثانية تستخدم فى نسخ ملف قاعدة البيانات كله أو جزء منه فى ملف آخر لايشترط أن يكون ملف قاعدة بيانات.

أ - الصورة الأولى

COPY TO <new filename>[<scope>][FIELDS <fieldlist>]
[WHILE <condition>] [FOR <condition>]

ويلاحظ أن جميع الإختيارات إختيارية ماعدا إسم الملف(<new filename>). وعند كتابة الأمر بدون باقى الإختيارات كالآتى مثلا :

COPY TO <new filename>

فإن هذا يؤدى إلى نسخ الملف المفتوح بالكامل فى الملف الجديد الذى يتم كتابة إسمه. أما إذا أريد نسخ حقول معينة أو سجلات معينة فقط فيتم استخدام الإختيارات الأخرى. وهذه الإختيارات تتلخص فى الآتى :

الإختيار (<scope>) ويستخدم لتحديد مدى محدد يراد البحث خلاله.

والإختيار (FIELDS<field list>) يستخدم لتحديد حقول معينة فقط يراد نسخها إلى الملف الجديد.

والإختيار (WHILE<condition>) يستخدم فى البحث عن سجلات تحقق شرطا أو شروطا معينة لنسخها فى الملف الجديد. وكذلك يستخدم الإختيار (FOR<condition>) لنفس الغرض.

ب - الصورة الثانية

COPY TO <new filename> [TYPE <file type>]

تستخدم هذه الصورة فى نسخ الملف المفتوح فى ملف آخر يمكن استخدامه بواسطة برامج أخرى غير برنامج (DBase III+). ويجب فى هذه الحالة إضافة الإمتداد الخاص بهذا الملف كما يجب كتابة نوع هذا الملف مكان الإختيار

(file type). فمثلا لنسخ ملف قاعدة البيانات إلى ملف يمكن استخدامه بواسطة برنامج لوتس ١-٢-٣ (Lotus 123) يتم كتابة السطر التالى :

```
COPY TO cadets.wks TYPE wks
```

كما يمكن نسخ مجموعة من الحقول فقط وكذلك مجموعة من السجلات عن طريق كتابة الإختيارات التى سبق شرحها فى الصورة الأولى.

٢١ - الأمـــر (COPY FILE)

يستخدم هذا الأمر لنسخ أى نوع من الملفات. والصورة العامة له كالآتى :

```
COPY FILE <file1> TO <file2>
```

ويجب ملاحظة أن أسماء الملفات هنا يجب أن تتضمن الإمتداد (Extension) ووحدة الأقراص الموجود عليها كل ملف كما يجب ملاحظة أن الملفات يجب ألا تكون مفتوحة. وعند نسخ ملف قاعدة بيانات (DBase file) يحتوى على حقل ملاحظات (memo) فيجب نسخ ملف الملاحظات المرتبط به. وهو الملف الذى يحتوى على الإمتداد (dbt.).

٢٢ - الأمـــر (COPY STRUCTURE)

يستخدم هذا الأمر فى نسخ تركيب ملف قاعدة البيانات فقط دون نسخ السجلات المخزنة به. والصورة العامة له كالآتى :

```
COPY STRUCTURE TO <filename>
        [FIELDS <field list>]
```

وإسم الملف يجب أن يشمل وحدة الأقراص الموجود عليها الملف إذا كانت غير وحدة الأقراص المبدئية (Default). ويستخدم الإختيار (FIELDS) فى تحديد حقول معينة يراد نسخها فى هذا التركيب.

٢٣ - الأمـــر (COPY STRUCTURE EXTENDED)

يستخدم هذا الأمر فى نسخ تركيب ملف قاعدة البيانات فى ملف آخر يحتوى على

أربعة حقول فقط وهى (Field Name) ، (Field Type) ، (Field Length) ، (Decimal Numbers).

ويتم إدخال حقول الملف المنسوخ إلى الملف الجديد كسجلات. وتستخدم هذه الطريقة فى تصميم البرامج التطبيقية التى يراد السماح للمستخدم بتعديل تركيب ملف قاعدة البيانات من خلالها دون الحاجة إلى استخدام الأمر (MODIFY STRUCTURE). حيث يتم تعديل تركيب الملف مثل تعديل أى سجل فى قاعدة البيانات. والصورة العامة للأمر كالآتى :

COPY TO <new file> STRUCTURE EXTENDED

وبعد السماح للمستخدم بإدخال التعديلات المطلوبة على هذا التركيب يتم إنشاء ملف قاعدة بيانات جديد من هذا التركيب باستخدام الأمر(CREATE FROM). فمثلا إذا كان هناك ملف للموظفين يحتوى على التركيب التالى :

Field	Field Name	Type	Width	Dec
1	name	character	30	
2	address	character	30	
3	phone	character	10	

فعند استخدام الأمر التالى :

COPY TO newname STRUCTURE EXTENDED

يصبح تركيب الملف كالآتى :

Field	Field Name	Type	Width	Dec
1	Field_name	Character	10	
2	Field_type	Character	1	
3	Field_len	Numeric	3	
4	Field_dec	Numeric	3	

وعند عرض سجلات هذا الملف بواسطة الأمر (List) مثلا يظهر الآتى :

Record	Field_Name	Field_Type	Width-Len	Field-dec
1	name	C	30	
2	address	C	30	
3	phone	C	10	

وفى هذه الحالة يمكن تعديل حقول الملف باستخدام أى أمر من أوامر التعديل مثل (Edit) ، (Change).

٢٤ - الأمـــــر (COUNT)

ويستخدم هذا الأمر فى حساب عدد السجلات التى تحقق شرطا أو شروطا معينة. والصورة العامة للأمر كالآتى :

```
COUNT[<Scope>][WHILE <codition>]
     [FOR <condition>][TO <memvar>]
```

ويلاحظ أن جميع الإختيارات إختيارية حيث يمكن كتابة الأمر (COUNT) دون كتابة أى شىء بعده. وفى هذه الحالة يتم حساب عدد السجلات فى ملف قاعدة البيانات المفتوح. أما إذا أريد حساب عدد السجلات التى تحقق شروطا معينة يتم استخدام الشروط الموجودة مع الأمر. كما يمكن استخدام متغير الذاكرة (memvar) فى تخزين هذا العدد لاستخدامه فى البرنامج حسب الحاجة.

فمثلا يمكن كتابة الأمر التالى :

```
COUNT FOR name = "Mohamed" TO mname
```

فى هذه الحالة يتم حساب عدد الأشخاص الذين يبدأ إسمهم بالإسم (Mohamed) ثم يتم تخزين هذا العدد فى المتغير (mname).

٢٥ - الأمـــــر (CREATE)

يستخدم هذا الأمر فى إنشاء ملف قاعدة البيانات وهو يؤدى إلى عرض الشاشة التى

تظهر عند استخدام قائمة الإنشاء (Create) فى القائمة الرئيسية لبرنامج المساعد (Assistant) واختيار (Database file). إرجع إلى الجزء الخاص ببرنامج المساعد (Assistant).

٢٦ - الأمـــر (CREATE FROM)

ويستخدم هذا الأمر فى إنشاء ملف قاعدة بيانات من ملف سبق نسخه بواسطة الأمر (COPY STRUCTURE EXTENDED).

والصورة العامة له كالآتى :

CREATE <new file> FROM <structure extended file >

إرجع إلى الأمر (COPY STRUCTURE EXTENDED).

٢٧ - الأمـــر (CREATE/MODIFY LABEL)

يستخدم هذا الأمر فى إنشاء العناوين البريدية (Labels) وهى صورة مصغرة من التقارير (Reports) تعطى معلومات سريعة عن سجلات قاعدة البيانات. والصورة العامة للأمر كالآتى :

CREATE/MODIFY LABEL <filename>/?

وهذا الأمر يؤدى إلى ظهور نفس الشاشة التى تظهر عند الدخول فى قائمة الإنشاء (Create) أو قائمة التعديل (Modify) واختيار (Label) مع ملاحظة أن الأمر (CREATE) يستخدم فى إنشاء التقرير والأمر (MODIFY) يستخدم فى إنشائه أو تعديله بعد ذلك. ويتم كتابة إسم الملف بدون الإمتداد حيث أن البرنامج يضيف الإمتداد (lbl.) إليه آليا. وإذا لم يتذكر المستخدم إسم الملف المطلوب فإنه يكتب (?) بدلا من إسم الملف وفى هذه الحالة تظهر قائمة بأسماء ملفات العناوين البريدية (Labels) الموجودة فى القرص أو فى الكتالوج المفتوح. ويحتوى عمود الإختيارات (Menu bar) الذى يظهر عند كتابة هذا الأمر على ثلاثة قوائم يمكن تلخيصها كالآتى :

١ - قائمة الإختيارات (Options)

وهى تحدد حجم التقرير (Size) كما يتضح من الشكل (٣٨ - ١١) وهناك خمسة أحجام قياسية يظهر أحدها على أول سطر فى القائمة وتظهر باقى الأحجام عند الضغط على مفتاح الإدخال. كما تستخدم باقى إختيارات القائمة فى تحديد عرض التقرير وارتفاعه والهامش الأيسر والمسافة بين التقارير وهكذا.

```
Options                              Contents                              Exit
 Predefined size:          3 1/2 x 15/16 by 1

 Label width:               35
 Label height:              5
 Left margin:               0
 Lines between labels:      1
 Spaces between labels:     0
 Labels across page:        1

 CURSOR    <-- -->  |          UP    DOWN | DELETE       | Insert Mode: Ins
  Char:     ←   →   | Record:   ↑     ↓   | Char:   Del  | Exit:        ^End
  Field: Home  End  | Page:   PgUp  PgDn  | Field:  ^Y   | Abort:       Esc
  Pan:     ^←  ^→   | Help:    F1         | Record: ^U   | Set Options:^Home
CREATE LABEL        ||<C:>||C:M.LBL              ||Opt: 1/7
```

شكل (٣٨ - ١١)

٢ - قائمة المحتويات (Contents)

ويتم عن طريقها تحديد محتويات التقرير أى أسماء الحقول أو متغيرات الذاكرة الممثلة لها. وعند كتابة أسماء الحقول يمكن الضغط على مفتاح (F10) لعرض قائمة بأسماء الحقول وإختيار الحقول المراد عرضها فى التقرير. ويمكن عرض أى عدد من السطور فى التقرير يحتوى كل منها على حقل أو أكثر. وعندما يراد عرض أكثر من حقل فى نفس السطر يتم فصلها بواسطة الفاصلة (,).

٣ - قائمة الخروج (Exit)

تستخدم هذه القائمة فى الخروج من قوائم العناوين البريدية (Label) وهى تحتوى على إختيارين :

أ - الإختيار (Save) ويستخدم فى تخزين التقرير الذى تم إنشاؤه أو التعديلات التى تم إدخالها على تقرير سابق. ويمكن تنفيذ هذه العملية أيضا بالضغط على مفتاحى (Ctrl-End).

ب- الاختيار (Abandon) ويستخدم فى الخروج من قوائم العناوين البريدية دون تخزين التقرير أو التعديلات التى تم إدخالها على تقرير سابق.

٢٨ - الأمـــر (CREATE/MODIFY QUERY)

يستخدم هذا الأمر فى إنشاء أو تعديل مرشح (Filter) لاستخدامه مع قاعدة البيانات. وهذا المرشح يؤدى نفس عمل المرشح بمفهومه الميكانيكى حيث أنه لايسمح بالمرور إلا للسجلات التى تحقق الشرط أو الشروط المطلوبة. والصورة العامة للأمر كالآتى :

CREATE/MODIFY QUERY <filename>/?

ويستخدم الأمر (CREATE) لإنشاء ملف مرشح جديد كما يستخدم الأمر (MODIFY) لتعديل ملف سبق إنشاؤه. ويتم كتابة إسم الملف بدون الإمتداد حيث أن البرنامج يضيفه آليا. وإذا لم يتذكر المستخدم إسم الملف المطلوب يمكنه كتابة الحرف (?) لعرض أسماء ملفات البحث المخزنة على القرص أو فى الكتالوج لاختيار الملف المطلوب تعديله.

وهذا الأمر يؤدى إلى ظهور نفس الشاشة التى تظهر عند الدخول فى قائمة الإنشاء (Create) أو قائمة التعديل (Modify) واختيار (Query). حيث يظهر عمود الإختيارات (Menu Bar) الذى يحتوى على القوائم التالية : انظر شكل (٣٨ - ١٢)

١ - القائمة (Set Filter)

وهذه القائمة تستخدم فى تحديد الشروط المطلوب إدخالها فى ملف البحث حيث تظهر عدة اختيارات تساعد المستخدم على إدخال هذه الشروط فى الجدول الذى يظهر على الشاشة ، وهذه الاختيارات تكون كالآتى :

- الإختيار الأول هو إسم الحقل (Field Name) الذى يؤدى إلى عرض قائمة بأسماء الحقول لاختيار الحقل المطلوب إدخاله فى الشرط.

```
Set Filter              Nest              Display              Exit
 Field Name
 Operator
 Constant/Expression
 Connect
 Line Number        1
```

Line	Field	Operator	Constant/Expression	Connect
1				
2				
3				
4				
5				
6				
7				

```
CREATE QUERY   |<C:>|C:M.QRY        |Opt: 1/2
```

شكل (٣٨ - ١٢)

- الإختيار الثانى هو معامل المقارنة (Operator) ويؤدى إلى ظهور قائمة بالمعاملات التى يتم اختيار معامل المقارنة المطلوب منها.
- الاختيار الثالث هو القيمة المطلوب مقارنتها (Constant/Expression) وهو يتيح للمستخدم كتابة القيمة المطلوب مقارنة محتويات الحقل الذى تم اختياره بها.
- الإختيار الرابع هو الربط (Connect) وهو يساعد المستخدم على الربط بين عدة شروط باستخدام المعاملات المنطقية مثل (AND) ، (OR).
- الإختيار الخامس هو رقم السطر (Line Number) وهو يساعد المستخدم على اختيار رقم سطر معين فى الجدول لتعديله. ويتغير هذا الرقم آليا عند الإنتقال من سطر إلى آخر.

ويمكن للمستخدم إضافة أى سطر بين السطور التى تمت كتابتها فى الجدول بالضغط على مفتاحى (Ctrl-N). كما يمكنه أيضا مسح أى سطر من السطور بالضغط على مفتاحى (Ctrl-U).

٢ - القائمة (Nest)

هذه القائمة تستخدم فى وضع الأقواس حول الشروط التى يراد وضعها داخل

الأقواس وذلك للتحكم فى ترتيب تنفيذ الشروط المختلفة. وهى تتيح للمستخدم إضافة أقواس أو حذف أقواس سبق إضافتها فى أى سطر من سطور الجدول. ويفيد ذلك فى تكوين العلاقات المركبة التى تتكون من عدة شروط.

٣ - القائمة (Display)

وتستخدم هذه القائمة فى عرض السجلات التى تحقق الشروط التى سبق إدخالها وذلك للتأكد أن الشروط قد تمت كتابتها بدقة. حيث يتم عرض بيانات أول سجل يحقق الشرط ثم يتم الإنتقال إلى السجلات التالية عن طريق الضغط على مفتاح (PgDn).

٤ - قائمة الخروج (EXIT)

وتستخدم لتخزين الملف الذى تم تكوينه أو الخروج دون تخزين الملف. إرجع إلى الجزء الخاص بإنشاء ملفات البحث عن طريق برنامج المساعد (Assistant).

٢٩ - الأمـــر (CREATE /MODIFY REPORT)

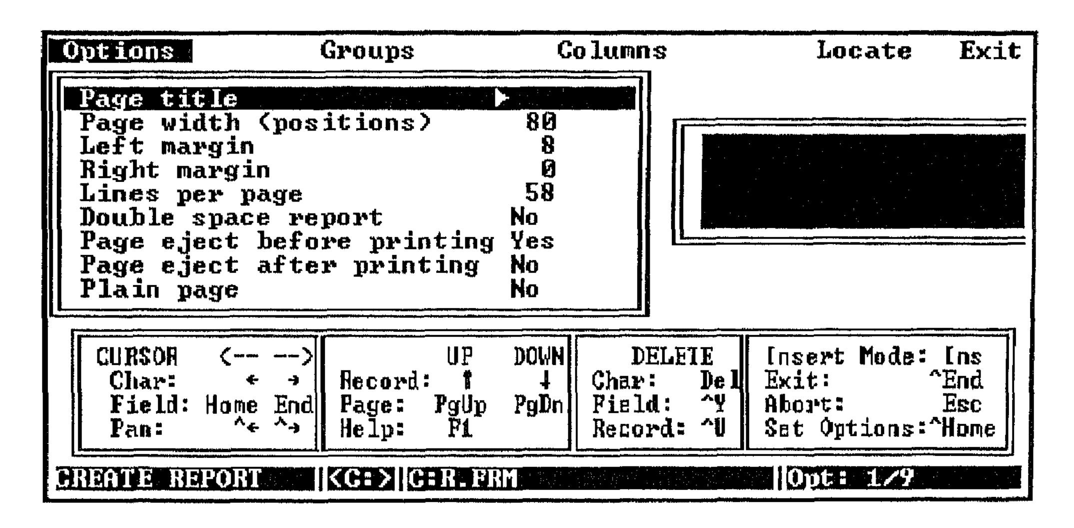
يستخدم هذا الأمر فى إنشاء التقارير (Reports) الجديدة أو تعديل تقارير سبق إنشاؤها وهذه التقارير يتم تصميمها على هيئة أعمدة تمثل الحقول المختلفة. والصورة العامة للأمر كالآتى :

CREATE/MODIFY REPORT <filename>/?

وهذا الأمر يؤدى إلى ظهور نفس الشاشة التى تظهر عند الدخول فى قائمة الإنشاء (CREATE) واختيار (REPORT). ويستخدم الأمر (CREATE) لإنشاء ملف تقارير جديد فى حين يستخدم الأمر (MODIFY) فى إنشاء أو تعديل ملف تقرير سبق إنشاؤه. ويتم إدخال إسم الملف بدون الإمتداد حيث أن البرنامج يضيف الإمتداد (FRM.) آليا. وإذا لم يتذكر المستخدم إسم الملف المطلوب تعديله فإنه يكتب الحرف (?) لعرض جميع الملفات المخزنة على القرص أو فى الكتالوج المفتوح ثم اختيار الملف المطلوب. وعند كتابة هذا الأمر يظهر عمود الإختيارات (Menu Bar) الذى يحتوى على مجموعة من الإختيارات يمكن تلخيصها كالآتى :

أ - الإختيارات (Options)

وهى تحتوى على عدة إختيارات يتم عن طريقها تحديد عنوان للصفحة (Page Title) بالإضافة إلى الإختيارات الخاصة بأبعاد الصفحة مثل العرض والهوامش وعدد السطور فى الصفحة و .... الخ. وعند إدخال عنوان الصفحة (Page Title) فإن البرنامج يتيح للمستخدم إدخال حتى (٤) سطور بحد أقصى (٦٠) حرفا. أما باقى الإختيارات فى القائمة فيمكن تعديلها حسب الحاجة ولكن فى الغالب تكون القيم المبدئية (Default) المكتوبة أمامها مناسبة. انظر الشكل (٣٨ - ١٣)



شكل (٣٨ - ١٣)

ب - المجموعات (Groups)

إستخدام المجموعات إختيارى وبعض التقارير لاتحتاج إلى استخدام هذا الإختيار ولكنها فى أحيان أخرى تكون مطلوبة لتحسين شكل التقرير والحصول على معلومات أكثر تفصيلا. حيث أن هذه المجموعات تتيح للمستخدم تقسيم بيانات التقرير بناء على البيانات الموجودة فى أحد الحقول. فمثلا إذا كانت هناك قاعدة بيانات للموظفين الذين يختلفون فى الوظائف (Jobs). فيمكن تصميم التقرير مع استخدام حقل الوظيفة (Job) كأساس للتقسيم وذلك عن طريق كتابة إسم هذا الحقل مكان الإختيار (Group on Expression) وعند طباعة التقرير تظهر

بيانات الموظفين الذين يشتركون فى الوظيفة فى مجموعة واحدة.

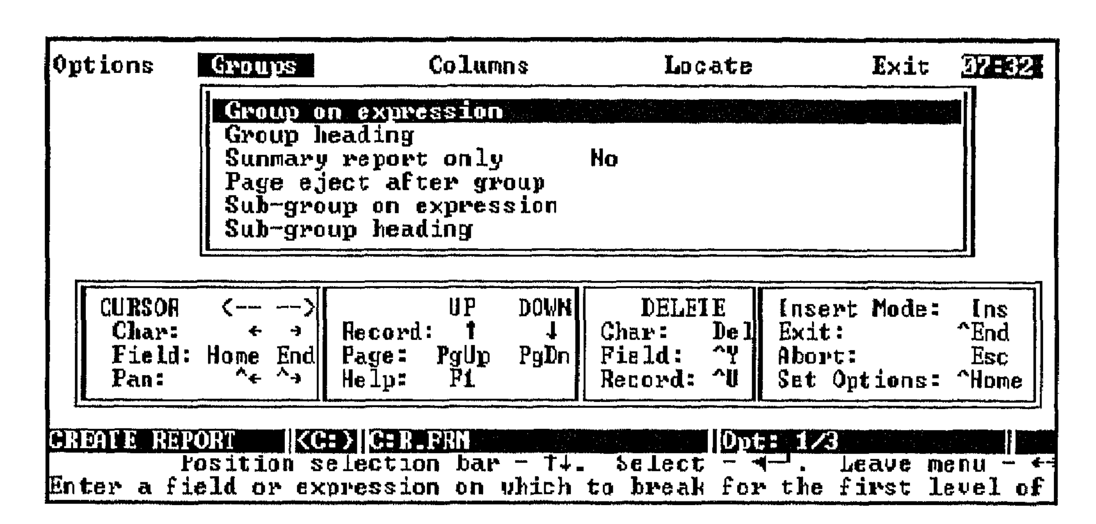
وإذا كانت هناك حقول عددية فى ملف قاعدة البيانات فيتم تجميع هذه البيانات لكل مجموعة على حدة. وهناك عدة اختيارات تحت قائمة المجموعات وهى كالآتى :

- الإختيار (Group on Expression) وهو يستخدم لإدخال إسم الحقل المراد التقسيم بناء عليه.
- الإختيار (Group Heading) ويستخدم لإدخال عنوان لكل مجموعة يوضح الحقل الذى تم التقسيم بناء عليه. فمثلا يمكن كتابة (Client's Job) كعنوان لكل مجموعة.
- الإختيار (Summary Report Only) وهو يستخدم لتحديد ما إذا كان المطلوب الحصول على تقرير مختصر أو تقرير تفصيلى. وفى حالة التقرير المختصر يتم عرض البيانات العددية لكل مجموعة بعد تجميعها دون عرض باقى البيانات. أما التقرير التفصيلى فيتضمن جميع بيانات السجلات الموجودة فى كل مجموعة.
- الإختيار (Page Eject After Group) ويستخدم لكتابة بيانات كل مجموعة فى صفحة منفصلة.
- الإختيار (Sub-group on Expression) ويستخدم لتجميع البيانات فى مجموعات فرعية داخل المجموعات ويتم ذلك بناء على حقل آخر يتم اختياره.
- الإختيار (Sub-group Heading) ويستخدم لتحديد عنوان لكل مجموعة فرعية. انظر شكل (٣٨ - ١٤)

جـ - الأعمدة (Columns)

وهذه القائمة تستخدم فى تحديد البيانات المطلوب كتابتها فى الأعمدة الخاصة بالتقرير مع تحديد مكان هذه البيانات وشكلها فى التقرير. وتحتوى هذه القائمة كما هو يتضح من الشكل (٣٨ - ١٥) على الإختيارات التالية :

- الإختيار (Contents) ويستخدم فى تحديد محتويات كل عمود من أعمدة التقرير. ويمكن أن يحتوى العمود على حقل بيانات (Field) أو متغير ذاكرة (Memory Variable) أو علاقة رياضية بين أكثر من حقل.



شكل (٣٨ - ١٤)

```
Options        Groups        Columns        Locate        Exit  07:40
                      Contents                nane
                      Heading                 nane
                      Width                   30
                      Decimal places
                      Total this column
──Report Format──────────────────────────────────────────
>>>>>>>>name
         XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
REATE REPORT  ||<C:>||C:R.FRM          ||Column: 1
 Position selection bar - ↑↓   Select - ◄─┘   Prev/Next column - PgUp/PgDn
```

شكل (٣٨ - ١٥)

- الإختيار (Heading) ويستخدم هذا الإختيار لكتابة عنوان للعمود.
- الإختيار (Width) ويستخدم هذا الإختيار لتحديد عرض العمود. وهو يكون نفس عرض الحقل الذى يتم إدخاله فى هذا العمود. وإذا كان العنوان (Heading) أطول من عرض الحقل فإن عرض العمود يمتد ليغطى عرض العنوان كما يمكن زيادة أو إنقاص عرض العمود بكتابة العرض المطلوب. وعند كتابة عرض أقل من عرض الحقل فإن البيانات التى تظهر فى هذا العمود

ينتقل جزء منها إلى السطر التالى.

- الإختيار (Decimal Places) ويستخدم مع الحقول العددية فقط وهو يحدد عدد الكسور العشرية المطلوب ظهورها فى التقرر لهذا العدد ، وهو يأخذ نفس القيمة التى سبق إدخالها عند إنشاء ملف قاعدة البيانات كما يمكن إدخال أى قيمة أخرى.
- الإختيار (Total This Column) ويستخدم مع الحقول العددية عندما يراد تجميع البيانات العددية الخاصة بهذا الحقل لمجموعة السجلات التى تظهر فى التقرير. والقيمة المبدئية لهذا الإختيار هى (Yes) وإذا أريد تغييرها يتم كتابة (N) أمام هذا الإختيار. وإذا أريد مسح العمود بعد إدخاله يتم الضغط على مفتاحى (Ctrl-U). وإذا أريد إضافة عمود قبل العمود الجارى إدخاله يتم الضغط على مفتاحى (Ctrl-N).

د - الإختيار (Locate)

يستخدم للوصول إلى أى عمود سبق إدخال بياناته حيث تظهر قائمة بأسماء الأعمدة التى سبق إدخالها فيتم اختيار العمود المطلوب تعديله. انظر الشكل (٣٨ - ١٦)

```
Options            Groups            Columns   Locate         Exit  03:13:50 pm
                                              ┌──────────────────────────┐
                                              │ name                     │
                                              │ AGE                      │
                                              └──────────────────────────┘

──Report Format──────────────────────────────────────────────────────────
>>>>>>>>name
                                              ──────────────────────────


        XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX ##

CREATE REPORT   |<C:>|C:R.FRM           |Opt: 1/2    |    |
```

شكل (٣٨ - ١٦)

هـ - الإختيار (Exit)

ويستخدم لتخزين التقرير الذى تم إنشاؤه أو التعديلات التى تم إدخالها أو

الخروج دون تخزين التقرير. إرجع إلى الجزء الخاص بإنشاء التقرير فى برنامج المساعد (Assistant).

٣٠ - الأمــــر (CREATE/MODIFY SCREEN)

يستخدم هذا الأمر فى إنشاء شاشات الإدخال الجديدة أو تعديل شاشات سبق إنشاؤها. وهو يؤدى إلى إنشاء ملفين أحدهما بالإمتداد (scr.) والآخر بالإمتداد (fmt.) حيث يساعد الملف ذو الإمتداد (scr.) على تعديل شاشة الإدخال ثم تخزينها فى الملف ذى الإمتداد (fmt.).

والصورة العامة للأمر كالآتى :

CREATE/MODIFY SCREEN <filename>/?

ويستخدم الأمر (CREATE) فى إنشاء ملف تشكيل (Format File) جديد كما يستخدم الأمر (MODIFY) فى إنشاء أو تعديل ملف تشكيل سبق إدخاله. ويكتب إسم الملف بدون الإمتداد حيث أن البرنامج يضيف إليه الإمتداد (scr.). وإذا لم يتذكر المستخدم إسم ملف التشكيل المطلوب تعديله يمكنه كتابة (?) بدلا من إسم الملف لعرض جميع ملفات التشكيل المخزنة على القرص أو فى الكتالوج المفتوح حيث يتم اختيار الملف المطلوب تعديله. وعند استخدام هذا الأمر تظهر الشاشة التى تظهر عند الدخول فى قائمة الإنشاء واختيار (Format For Screen).

تحــذير

فى حالة مسح الملف ذى الإمتداد (scr.) لايمكن تعديل ملف التشكيل ذى الإمتداد (fmt.) المقابل له حيث أن ملف التشكيل ذى الإمتداد (fmt.) لايمكن ترجمته إلى ملف الشاشة ذى الإمتداد (scr.). ولكن فى هذه الحالة يمكن تعديله كملف برنامج باستخدام الأمر (MODIFY COMMAND). وعند تعديل ملف التشكيل ذى الإمتداد (fmt.) بواسطة الأمر (MODIFY COMMAND) فلا يمكن تعديله بعد ذلك باستخدام الأمر (MODIFY SCREEN).

ولمزيد من المعلومات عن هذا الأمر إرجع إلى الجزء الخاص بإنشاء شاشة إدخال البيانات من خلال برنامج المساعد (Assistant).

٣١ - الأمـــر (CREATE/MODIFY VIEW)

يستخدم هذا الأمر فى ربط عدة ملفات قواعد بيانات عن طريق ملف واحد يسمى ملف المنظر (View File) وهذا الملف يحتوى على حقول يتم اختيارها من عدة ملفات. ويستخدم هذا الملف الجديد فى عرض أى بيانات من هذه الملفات أو تعديلها. والصورة العامة للأمر كالآتى :

CREATE/MODIFY VIEW <filename>/?

ويستخدم الأمر (CREATE) فى إنشاء ملف المنظر (View File) كما يستخدم الأمر (MODIFY) فى إنشاء أو تعديل أى ملف منظر سبق إنشاؤه. ويتم كتابة إسم الملف بدون الإمتداد حيث أن البرنامج يضيف الإمتداد (VUE.) آليا وإذا لم يتذكر المستخدم إسم الملف المطلوب تعديله يمكنه كتابة الحرف (?) مكان إسم الملف فتظهر قائمة بأسماء ملفات المنظر الموجودة على القرص أو فى الكتالوج المفتوح.

وعند تخزين ملف المنظر الذى تم إنشاؤه يصبح الملف مفتوحا كما يمكن فتح أى ملف منظر بكتابة الأمر (<SET VIEW TO <filename). كما يمكن إغلاق أى ملف منظر مفتوح بكتابة الأمر (CLOSE DATABASES) وهذا يؤدى إلى إغلاق جميع ملفات قواعد البيانات والملفات المرتبطة بها مثل ملفات الفهرس والتشكيل .... الخ كما يؤدى إلى إغلاق ملف المنظر الذى يربط هذه الملفات. وعند كتابة هذا الأمر يظهر عمود الإختيارات (Menu Bar) الذى يحتوى على الإختيارات التالية :

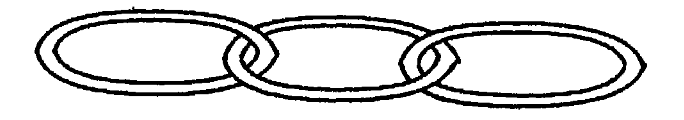
١ - الإختيار (Set Up)

وهذا الإختيار يعرض على الشاشة قائمة بأسماء ملفات قواعد البيانات الموجودة على القرص أو فى الكتالوج المفتوح حتى يستطيع المستخدم اختيار الملفات المطلوب إدخالها فى ملف المنظر. ولاختيار أى ملف من الملفات يتم تحريك العمود الضوئى (Highlight) إلى إسم هذا الملف ثم الضغط على مفتاح الإدخال وهذا يؤدى إلى ظهور مثلث يسار إسم الملف. كما يمكن إزالة هذا المثلث بالضغط على مفتاح الإدخال مرة ثانية وهذا يعنى عدم اختيار هذا الملف. وكل ملف يتم اختياره يتم اختيار ملف الفهرس الخاص به. ويمكن إدخال حتى تسعة ملفات قواعد بيانات والملفات الملحقة بها فى ملف منظر واحد.

٢ - الإختيار (Relate)

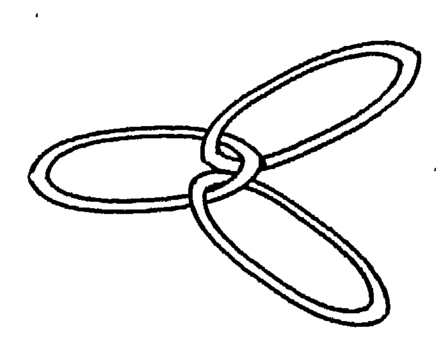
ويستخدم فى تحديد الحقول التى يتم عن طريقها ربط الملفات ببعضها. ويتم عن طريق هذه القائمة عرض الملفات التى سبق اختيارها حتى يتم فتح الملفات التى سوف تستخدم فى تكوين سلسلة العلاقة (Relation Chain). وترتيب فتح هذه الملفات مهم جدا حيث أن أول ملف يتم فتحه يعتبر بداية السلسة (Chain).

وبعد فتح هذا الملف تظهر قائمة بباقى الملفات التى سبق اختيارها من خلال قائمة التجهيز (Set Up) فيتم تحديد الملف المطلوب ربطه بالملف الأول. ثم يسأل البرنامج عن إسم الحقل الذى يتم الربط بناء عليه فيتم اختيار إسم هذا الحقل. وبنفس الطريقة يمكن ربط باقى الملفات عندما يراد ربط أكثر من ملفين فى نفس السلسلة (Chain). ويجب ملاحظة أن الربط يتم على التوالى فلا يجوز مثلا ربط ملف جديد بملف سبق ربطه بملفات تالية أى أن السلسلة يجب أن تكون على الشكل التالى :



شكل (٣٨ - ١٧)

وليست كالشكل التالى :



شكل (٣٨ - ١٨)

ويمكن ربط الملفين من خلال مشيرة النقطة (Dot Prompt) أو من خلال البرنامج عن طريق كتابة السطور التالية :

```
SELECT 1
USE file1
SELECT 2
USE file2
SELECT 1
SET RELATION TO field1 INTO file2
```

حيث (field1) هو الحقل المشترك بين الملفين (file1) و (file2).

٣ - الإختيار (Set Fields)

وهذا الإختيار يسمح للمستخدم باختيار أسماء الحقول المطلوب إدخالها فى ملف المنظر. وفى هذه الحالة يتم اختيار كل ملف من الملفات المرتبطة فى العلاقة فتظهر أسماء الحقول الخاصة بهذا الملف ويتم اختيار جميع الحقول المطلوبة. ويلاحظ أن الحقول تظهر فى البداية وأمامها علامة (◀) دليل على اختيارها. ولإلغاء اختيار أى حقل يتم الضغط على مفتاح الإدخال عند وقوف المؤشر على إسم هذا الحقل.

٤ - الإختيار (Options)

وهى قائمة تحتوى على الإختيار (Format) الذى يستخدم فى تعديل ملف التشكيل (Format File) الذى سبق إنشاؤه للملف المنظر. وإذا أريد إنشاء ملف تشكيل جديد يتم إنشاؤه من خلال قائمة الإنشاء (Create) كما سبق الإيضاح. وتحتوى هذه القائمة أيضا على الإختيار (Filter) وهو يتيح للمستخدم إدخال الشروط المطلوب إدخالها للبحث عن السجلات.

٥ - الإختيار (Exit)

وهو يتيح للمستخدم تخزين ملف المنظر الذى تم إنشاؤه من خلال الإختيار (Save) أو الخروج دون تخزينه (Abandon).

٣٢ - الأمــــر (CREATE VIEW FROM ENVIRONMENT)

يستخدم هذا الأمر فى إنشاء ملف منظر (View File) من العلاقات (Relations) التى سبق إنشاؤها من خلال أوامر البرنامج. ويفيد هذا الأمر فى ربط الملفات وإنشاء ملف منظر لها من خلال البرنامج دون الرجوع إلى برنامج المساعد (Assistant) والصورة العامة لهذا الأمر كالآتى :

CREATE VIEW <filename> FROM ENVIRONMENT

وعند كتابة هذا الأمر يتم تخزين جميع ملفات قواعد البيانات وملفات الفهرس والعلاقات التى تم إنشاؤها بينها عن طريق الأمر (SET RELATION TO). كما يتم إدخال جميع الحقول الخاصة بهذه الملفات فى ملف المنظر وذلك فى حالة عدم كتابة الأمر (SET FIELDS TO) الذى يمكن من خلاله تحديد حقول معينة.

٣٣ - الأمــــر (DELETE)

يستخدم هذا الأمر فى تحديد السجلات المطلوب مسحها. وهو يؤدى نفس العمل الذى يتم تنفيذه فى حالة استخدام برنامج المساعد (Assistant) والدخول فى قائمة التحديث (Update) واختيار (Delete). والصورة العامة للأمر كالآتى :

DELETE [<scope>] [WHILE <condition>] [FOR <condition>]

ويلاحظ هنا أن جميع الإختيارات إختيارية أى يمكن كتابة الأمر دون كتابة أى شىء بعده. وفى هذه الحالة يتم مسح السجل الذى يقف عنده مؤشر السجلات (Record Pointer) فى هذا الوقت. ويجب ملاحظة أن هذا الأمر لايمسح السجلات مباشرة ولكنه يضع علامة عندها حتى يتم مسحها نهائيا باستخدام الأمر (PACK). ويلاحظ عند عرض السجلات باستخدام الأمر (DISPLAY) والأمر (LIST) ظهور علامة (*) أمام كل سجل سبق مسحه باستخدام الأمر (DELETE) وهذا يعنى أن السجل موجود ولكنه جاهز للمسح. والإختيار (scope) يستخدم لتحديد المدى الذى يتم البحث خلاله عن السجلات المطلوب مسحها. كما يستخدم الإختيار (WHILE <condition>) لتحديد الشرط أو الشروط التى يتم البحث بناء عليها وكذلك يستخدم الإختيار(FOR <condition>) لنفس الغرض.

مثـال

لتجهيز أول سجل فى قاعدة البيانات (Cadets) للمسح يتم كتابة الأوامر التالية :

```
. USE Cadets
. DELETE
1 record deleted
```

يلاحظ ظهور الرسالة الدالة على تجهيز سجل واحد للمسح. كما يمكن كتابة السطر التالى لتجهيز السجل رقم ٨ للمسح.

```
. DELETE RECORD 8
 1 record deleted
```

ويجب ملاحظة أن (RECORD 8) هنا تمثل المدى (Scope). كما يمكن كتابة السطر التالى لمسح جميع السجلات التى يبدأ حقل الإسم فيها بالإسم (Mohamed) مثلا :

```
DELETE ALL FOR name = 'Mohamed'
```

ويجب ملاحظة أن (ALL) هنا تمثل المدى (scope) حيث يتم البحث خلال ملف قاعدة البيانات كله.

٣٤ - الأمـــــر (DIR)

يستخدم هذا الأمر فى عرض دليل ملفات قواعد البيانات الموجودة على وحدة الأقراص المستخدمة أو الفهرس الفرعى المستخدم. والصورة العامة له كالآتى :

```
DIR [<drive:>] [<path>]
```

وكتابة الأمر دون كتابة أى شىء بعده تؤدى إلى ظهور أسماء ملفات قواعد البيانات الموجودة على وحدة الأقراص الحالية أو الفهرس الفرعى الحالى. كما يمكن استخدام هذا

الأمر فى عرض جميع الملفات عن طريق إضافة الحروف الشاملة (Global Characters) وذلك كالآتى مثلا :

DIR *.*

وهذا يؤدى إلى عرض جميع الملفات الموجودة فى وحدة الأقراص أو الفهرس الحالى سواء كانت ملفات قواعد بيانات أو أى ملفات أخرى. كما يمكن عرض ملفات الفهرس فقط عن طريق كتابة السطر التالى :

DIR *.NDX

كما يمكن استخدام المسار (Path) كالآتى مثلا :

DIR SALES *.*

ويؤدى هذا إلى عرض جميع الملفات الموجودة فى الفهرس الفرعى (SALES). كما يمكن عرض ملفات قواعد البيانات الموجودة فى هذا المسار كالآتى :

DIR SALES\

٣٥ - الأمـــر (DISPLAY)

يستخدم هذا الأمر فى عرض بيانات السجلات المطلوبة. والصورة العامة له كالآتى :

DISPLAY [<scope>][<expression list>]
[WHILE <condition>][FOR <condition>][OFF]
[TO PRINT]

ويلاحظ هنا أن جميع الإختيارات إختيارية حيث يمكن كتابة الأمر دون كتابة أى شيء بعده. وفى هذه الحالة تظهر بيانات السجل الذى يقف عنده مؤشر السجلات (Record Pointer) كما يتم عرض جميع الحقول الخاصة بهذا السجل.

وعند زيادة بيانات الحقول عن عرض الشاشة تنتقل البيانات إلى السطر التالى.

وعند عرض عدد من السجلات يزيد عن طول الشاشة فإن عرض السجلات يتوقف مؤقتا (Pause) حتى يضغط المستخدم على أى مفتاح لاستكمال العرض.

والإختيار (<scope>) يستخدم فى تحديد المدى المطلوب البحث خلاله.

والإختيار (<expression list>) يستخدم فى تحديد الحقول المطلوب عرض بياناتها.

والإختيار (WHILE <condition>) يستخدم فى تحديد السجلات المطلوب عرض بياناتها والتى تحقق الشرط أو الشروط المكتوبة. وكذلك الإختيار(FOR <condition>) يؤدى نفس الغرض. ولكن الإختيار (WHILE) أقوى وأسرع فى الوصول إلى السجلات المطلوبة.

والإختيار (OFF) يستخدم عندما يراد عدم ظهور أرقام السجلات على الشاشة.

والإختيار (TO PRINT) يستخدم عندما يراد طباعة البيانات المعروضة ويجب ملاحظة أن أسماء الحقول تظهر فوق البيانات المعروضة. وعندما يراد عدم عرض أسماء الحقول يستخدم الأمر (SET HEADING OFF). كما أن أسماء الحقول تظهر بالحالة التى يتم كتابتها بها فى الأمر أى إذا تمت كتابتها بحروف كبيرة (Capital) تظهر على الشاشة بحروف كبيرة أيضا. فمثلا لعرض حقول الإسم والعنوان فى ملف الطلبة (Cadets) يتم كتابة السطور التالية :

```
USE CADETS
DISPLAY NEXT 3 NAME, ADDRESS
```

وفى هذه الحالة تظهر الشاشة التالية :

Record #	NAME	ADDRESS
1	MOHAMED	12 - Ahram Street
2	TAREK	8 - Tayaran Street
3	GALAL	20 - Gomhorya Square

ويلاحظ هنا أن أسماء الحقول فى رأس القائمة (Heading) مكتوبة بحروف كبيرة

لأنها تم كتابتها فى الأمر بحروف كبيرة (Capital).

ملاحـظة

حقول الملاحظات (memo fields) لاتظهر محتوياتها على الشاشة إلا عند كتابتها منفصلة فى الأمر وفى هذه الحالة تظهر محتويات هذا الحقل فقط على الشاشة. فمثلا إذا كان هناك حقل ملاحظات يسمى (Notes) فلكى يتم عرض البيانات المخزنة فيه بالنسبة للسجل الذى يقف عنده مؤشر السجلات (Record Pointer) يتم كتابة الأمر التالى :

DISPLAY Notes

وفى هذه الحالة تظهر الملاحظات الخاصة بهذا السجل بعرض (٥٠) حرفا فى السطر وإذا أريد تغيير هذا العرض يستخدم الأمر (SET MEMOWIDTH) فى تحديد طول السطر فى حقل الملاحظات.

٣٦ - الأمـــــر (DISPLAY HISTORY)

يستخدم هذا الأمر فى عرض آخر عشرين أمرا تم تنفيذها والصورة العامة له كالآتى :

DISPLAY HISTORY [LAST <N>] [TO PRINT]

والإختيار (LAST<N>) يستخدم لتحديد عدد الأوامر المطلوب عرضها على الشاشة. وهذا العدد لايزيد عن عشرين أمرا حيث أنه هو العدد المبدئى (Default) كما يمكن زيادة هذا العدد عن طريق الأمر (SET HISTORY TO).

والإختيار (TO PRINT) يستخدم عندما يراد طباعة هذه الأوامر على الطابعة.

٣٧ - الأمـــــر (DISPLAY MEMORY)

يستخدم هذا الأمر فى عرض حالة متغيرات الذاكرة (Memory Variables) الموجودة فى الذاكرة المؤقتة (RAM) فى أى وقت. حيث يعرض إسم كل متغير ونوعه وحجمه. كما يعرض أيضا بيانات عن حجم الذاكرة المستخدمة وحجم الذاكرة المتاحة.

والصورة العامة لهذا الأمر كالآتى :

DISPLAY MEMORY [TO PRINT]

ويلاحظ هنا وجود إختيار واحد فقط وهو (TO PRINT) وهو يستخدم عندما يراد طباعة هذه البيانات على الطابعة. ويمكن استخدام حتى (٢٥٦) متغير ذاكرة بحد أقصى (٦٠٠٠) حرف كما يمكن زيادة هذا العدد من خلال ملف المواصفات (Config.sys). وعندما تزيد البيانات المعروضة عن طول الشاشة يتوقف العرض (Pause) وتظهر الرسالة "Press any key to continue " .

٣٨ - الأمـــر (DISPLAY STATUS)

يستخدم هذا الأمر فى عرض بيانات عن الحالة الحالية للبرنامج. والصورة العامة له كالآتى :

DISPLAY STATUS [TO PRINT]

وهذا يؤدى إلى عرض بيانات عن ملفات قاعدة البيانات المفتوحة ورقم منطقة العمل (Work Area) المستخدمة والعلاقات المستخدمة بين الملفات إن وجدت وملفات الفهرس المفتوحة و ... الخ.

٣٩ - الأمـــر (DISPLAY STRUCTURE)

يستخدم هذا الأمر فى عرض بيانات عن ملف قاعدة البيانات المفتوح وتركيبه (Structure) الذى يشمل إسم كل حقل ونوعه وعرضه. كما يعرض عدد الحروف (Bytes) التى يتكون منها السجل الواحد. والصورة العامة لهذا الأمر كالآتى :

DISPLAY STRUCTURE [TO PRINT]

وينتج العدد الكلى للحروف (Bytes) فى السجل الواحد من مجموع عدد الحروف فى جميع الحقول المكونة لهذا السجل بالإضافة إلى حرف آخر يتم حجزه لعلامة المسح التى توضع أمام السجل عندما يراد تجهيزه للمسح.

وعند زيادة عدد سطور السجل عن طول الشاشة يتوقف عرض السطور وتظهر الرسالة "Press any key to continue". ثم يتم إستكمال عرض بيانات السجل عند ضغط المستخدم على أى مفتاح.

٤٠ - الأمــــر (DO)

يستخدم هذا الأمر فى تنفيذ برنامج فرعى (Module) أو برنامج خطوات (Procedure) كما يسمح بإدخال المعاملات (Parameters) المطلوبة فى البرنامج. والصورة العامة للأمر كالآتى :

DO <filename> [WITH <parameter list>]

ويجب أن يتضمن إسم الملف (filename) رمز وحدة الأقراص أو الفهرس الفرعى الذى يحتوى على ملف البرنامج المطلوب. ولايتم كتابة الإمتداد فى إسم الملف حيث أن البرنامج يضيف الإمتداد (.prg) آليا.

والإختيار (WITH <parameter list>) يستخدم لإدخال معاملات(Parameters) إلى البرنامج الفرعى الذى يتم تنفيذه. وعندما ينتهى تنفيذ البرنامج الذى تم استدعاؤه بواسطة الأمر (DO) ينتقل التحكم إلى البرنامج الذى قام باستدعائه.

ملاحـظة

عدد البرامج التى يتم إستدعاؤها بواسطة الأمر (DO) يحسب ضمن عدد الملفات المسموح بفتحها فى نفس الوقت. ولكن فى حالة استخدام ملف الخطوات أو الإجراءات (Procedure File) فإن فتح هذا الملف يحسب كملف واحد مفتوح بالرغم من أنه يحتوى على العديد من البرامج التى يمكن تنفيذها بواسطة الأمر (DO). لذلك يفضل استخدام ملف الخطوات أو الإجراءات (Procedure File) على استخدام البرامج المنفصلة (Modules) فى أغلب الأحيان.

مــثال ١

لتنفيذ البرنامج (Cadrep) من داخل البرنامج (Cadets) مثلا يتم كتابة الأمر التالى :

```
DO B: Cadrep
```

مثال ٢

لكتابة البرنامج (Calc) الذى يحسب مساحة المستطيل ، فإن هذا البرنامج يكون كالآتى مثلا :

```
PARAMETERS Length, Width, Area
Area = Length * Width
RETURN
```

ولكى يتم تنفيذ هذا البرنامج مع إدخال الأطوال 8 ، 5 مكان المعاملات (Length)، (Width) على الترتيب يتم كتابة السطور التالية :

```
Area = 0
DO Calc WITH 8,5 , Area
? Area
```

وعند الضغط على مفتاح الإدخال يظهر العدد (40). والسطر الأول يتم كتابته لحجز مكان فى الذاكرة للمتغير (Area).

٤١ - الأمـــر (DO CASE)

يستخدم هذا الأمر فى الإنتقال بين مجموعات مختلفة من الأوامر بناء على شروط معينة يتم إدخالها. وهو يعتبر صورة مكبرة من الأمر (IF) حيث أن الأمر (IF) يسمح بالإختيار بين بديلين فقط أما هذا الأمر فيسمح بالإختيار بين عدة بدائل. والصورة العامة له كالآتى :

```
DO CASE
        CASE <condition>
         <commands>
        CASE <condition>
```

```
        <commands>
    [OTHERWISE]
        <commands>
    ENDCASE
```

ويلاحظ أنه يتم التفرع إلى أى حالة (CASE) عندما يصبح الشرط (Condition) الخاص بها صحيحا (True). والشرط (Condition) هو عبارة عن علاقة منطقية (Logical) مثل (A = B) أو (mchoice = 1). ويبدأ البرنامج باختبار الشرط الأول فإذا تحقق هذا الشرط يتم تنفيذ الأوامر (Commands) التالية له وإذا لم يتحقق ينتقل البرنامج إلى الشرط الثانى .... وهكذا. وعندما ينفذ البرنامج مجموعة من الأوامر التابعة لحالة معينة (CASE) فإنه لايقوم باختبار باقى الحالات وإنما ينتقل إلى الأوامر التالية للأمر (ENDCASE).

والإختيار (OTHERWISE) إختيارى يمكن عدم كتابته ويستخدم عندما يراد تنفيذ مجموعة من الأوامر فى حالة عدم تحقق أى شرط من الشروط السابقة.

٤٢ - الأمـــــر (DO WHILE)

يستخدم هذا الأمر فى تكوين حلقة تكرارية (Loop) يتم تنفيذها أى عدد من المرات طالما كان الشرط الموجود بعد الأمر صحيحا (True). والصورة العامة لهذا الأمر كالآتى :

```
    DO WHILE <condition>
      <commands>
    ENDDO
```

ويستخدم الأمر (ENDDO) لتحديد نهاية الحلقة حيث يعود البرنامج إلى أول الحلقة ليختبر الشرط مرة ثانية. فإذا تحقق الشرط يتم تنفيذ أوامر الحلقة وإذا لم يتحقق يتم الإنتقال إلى الأوامر التى تلى الأمر (ENDDO). ويمكن استخدام حلقات متداخلة (Nested). وفى هذه الحالة يجب التأكد من ملاحظة إنتهاء كل حلقة داخل الحلقة الخارجية. ولتنظيم هذه العملية يمكن كتابة ملاحظات (Comments) بجوار كل أمر (ENDDO) لتوضيح الحلقة التى يتبعها مع ملاحظة أن أى تعليق (Comment) بجوار الأمر (ENDDO) يتم إهماله بواسطة البرنامج أى أنه يفيد فى التوضيح فقط فى

حالة عرض أوامر البرنامج.

ويمكن استخدام التعويض أو الإحلال بالماكرو (Macro Substitution) ولكن يراعى فى هذه الحالة عدم تغيير قيمة هذا الماكرو داخل الحلقة التكرارية.

والبرنامج التالى مثلا يوضح هذه الطريقة.

```
USE Cadets INDEX Name
STORE (UPPER(name) = 'TAREK') TO condition
DO WHILE & condition .AND. .NOT. EOF()
   ? TRIM(name), address
   SKIP
ENDDO
USE
RETURN
```

وهذا البرنامج يقوم أولا بتخزين الشرط الموضح فى متغير ذاكرة إسمه (Condition) ثم يقوم باستخدام هذا المتغير فى الشرط الموجود بعد الأمر (DO WHILE).

٤٣ - الأمـــــر (EDIT)

يستخدم هذا الأمر فى تعديل بيانات سجل معين فى ملف قاعدة البيانات. وهو يؤدى نفس العمل الذى يتم عند استخدام برنامج المساعد (Assistant) عن طريق الدخول إلى قائمة التحديث (Update) واختيار (EDIT). والصورة العامة لهذا الأمر كالآتى :

```
EDIT[<scope>][FIELDS <list>][WHILE <condition>]
     [FOR <condition]
```

ويلاحظ أن جميع الإختيارات إختيارية أى يمكن عدم استخدامها. وفى حالة كتابة الأمر (EDIT) دون كتابة أى شىء بعده تظهر البيانات الخاصة بالسجل الذى يقف عنده مؤشر السجلات (Record Pointer). وفى هذه الحالة يمكن تعديل البيانات عن طريق تحريك مؤشر التصحيح إلى الحقل المطلوب تعديله وكتابة البيانات الجديدة. ويتم الإنتقال من سجل إلى آخر باستخدام مفتاحى (PgUp) ، (PgDn). كما يتم تخزين التعديلات

التى تم إدخالها بالضغط على مفتاحى (Ctrl-End).

ولتعديل حقل الملاحظات (Memo Field) يتم وضع مؤشر التصحيح على هذا الحقل. ثم بالضغط على مفتاحى (Ctrl-PgDn) يتم فتح حقل الملاحظات حيث يتم تعديل الملاحظات أو إضافة ملاحظات جديدة. كما يتم تخزين هذه الملاحظات بالضغط على مفتاحى (Ctrl-PgUp) أو مفتاحى (Ctrl-End).

٤٤ - الأمـــــر (EJECT)

يستخدم هذا الأمر فى إرسال شفرة نقل الصفحة (Form Feed) إلى الطابعة وهى عبارة عن شفرة الآسكى (ASCII 12). ويراعى فى بداية الطباعة أن تبدأ الورقة من أولها. والصورة العامة للأمر كالآتى :

```
EJECT
```

وهذا الأمر يعيد عداد السطور (Prow()) وعداد الأعمدة (Pcol()) إلى الصفر.

٤٥ - الأمـــــر (ERASE)

يستخدم هذا الأمر فى مسح ملف من القرص أو من الفهرس الفرعى والصورة العامة له كالآتى :

```
ERASE <filename>/?
```

وإسم الملف يجب أن يتضمن الإمتداد كما يجب كتابة وحدة الأقراص أو الفهرس الفرعى فى حالة عدم وجود الملف المطلوب مسحه على وحدة الأقراص الحالية (Current Drive). وهذا الأمر يقوم بتنفيذ نفس العمل الذى يتم عند استخدام برنامج المساعد (Assistant) والدخول فى قائمة الأدوات (Tools) ثم اختيار (ERASE). كما أن استخدام هذا الأمر يماثل تماما استخدام الأمر (DELETE FILE). ويجب ملاحظة أن هذا الأمر لايسمح باستخدام الحروف الشاملة لمسح عدد من الملفات مثل أوامر نظام التشغيل (MS-DOS).

٤٦ - الأمـــر (EXIT)

يستخدم هذا الأمر فى الخروج من الحلقة التكرارية التى تتكون نتيجة لاستخدام الأمر (DO WHILE). ويؤدى هذا الأمر إلى الإنتقال إلى الأوامر التى تلى الأمر (ENDDO). والصورة العامة له كالآتى :

```
EXIT
```

٤٧ - الأمـــر (FIND)

يستخدم هذا الأمر للبحث فى ملف قاعدة البيانات المفهرس (Indexed) عن أول سجل يماثل الحقل الفهرسى فيه سلسلة حرفية أو عدد معين يتم إدخاله. وهو يوفر سرعة بحث عالية جدا. والصورة العامة له كالآتى :

```
FIND <character string> /<n>
```

وإذا لم تكن السلسلة الحرفية أو العدد الذى يجرى البحث عنه موجودا فى قاعدة البيانات تظهر العبارة "No found" على الشاشة وفى هذه الحالة يصبح مؤشر السجلات (Record Pointer) عند آخر الملف أى أن الدالة (EOF()) تصبح صحيحة (True). وعادة لاتوضع السلسلة الحرفية التى يتم البحث عنها بين علامات تنصيص (Quotation Marks) ولكن إذا كانت السلسلة الحرفية تحتوى على مسافات خالية فى أولها فيجب فى هذه الحالة وضعها بين علامات تنصيص. ويجب ملاحظة أن المقارنة تصبح صحيحة إذا كانت الحروف الأولى فى السلسلة الحرفية المطلوبة مطابقة للحروف الأولى من حقل المفتاح (Index Key) بصرف النظر عن باقى الحروف.

وعند البحث عن متغير ذاكرة حرفى يجب إضافة الدالة (&) قبل إسم المتغير. وإذا كان هذا المتغير يحتوى على مسافات فى بدايته يتم وضع الإسم مع الدالة (&) بين علامات تنصيص (Quotation Marks) كالآتى مثلا " &mname" . .

والأمر (FIND) يماثل الأمر (SEEK) ولكن الأمر (SEEK) أشمل منه حيث يمكن استخدام العلاقات الحسابية معه كما سبق الإيضاح فى الجزء الثانى من الكتاب.

مـثـال

للبحث عن أى إسم يبدأ بالحرف (M) فى ملف الطلبة (Cadets) يتم كتابة السطور التالية :

```
. USE Cadets INDEX Name
. FIND M
. ? name
Mohamed
```

ويلاحظ فى هذه الحالة ظهور الإسم (Mohamed) عند السؤال عن الإسم الموجود فى الحقل الفهرسى (Name) باستخدام الأمر (?). ويلاحظ أن المقارنة قد تمت بين الحرف المطلوب والحرف الأول فى حقل الإسم فقط. ولكن فى حالة استخدام الأمر (SET EXACT ON) يصبح الأمر مختلفا. فمثلا عند كتابة الأوامر السابقة بالصورة الآتية :

```
. USE Cadets INDEX Name
. SET EXACT ON
. FIND M
No found
```

ويلاحظ فى هذه الحالة ظهور العبارة "No found" وذلك لعدم وجود الإسم المطابق تماما للحرف (M) أى الذى يتكون من الحرف (M) فقط.

٤٨ - الأمـــر (GO/GOTO)

يستخدم هذا الأمر فى توجيه مؤشر السجلات (Record Pointer) إلى سجل معين. وهناك صورتان لهذا الأمر :

الصورة الأولى تكون كالآتى :

```
[GO/GOTO] <N>
```

حيث (N) هو رقم السجل المراد توجيه المؤشر إليه. ويلاحظ أن الأمر (GO/GOTO) إختيارى فى هذه الحالة. أى يكفى كتابة الرقم فقط دون كتابة الأمر كالآتى مثلا :

```
.  USE Cadets
   8
. ? RECNO()
   8
```

ويلاحظ هنا عند السؤال عن رقم السجل الذى يقف عنده المؤشر باستخدام الدالة (RECNO()) ظهور الرقم (٨).

كما يمكن استخدام متغيرات الذاكرة مع الأمر (GO/GOTO) وذلك كالآتى مثلا :

```
. STORE 5 TO mnum
. GO mnum
. ? RECNO()
  5
```

والصورة الثانية تكون كالآتى :

GO/GOTO BOTTOM/TOP

ويلاحظ هنا أن الأمر (GO/GOTO) ليس إختياريا كالحالة السابقة أى أن وجوده ضرورى فى هذه الحالة.

مثال

```
.  GO TOP
.  ? RECNO()
   1
```

ويلاحظ ظهور الرقم (1) عند السؤال عن رقم السجل.

٤٩ - الأمـــر (HELP)

يستخدم هذا الأمر فى عرض شاشات المساعدة التى يتم من خلالها شرح أوامر البرنامج. والصورة العامة للأمر كالآتى :

```
HELP keyward
```

حيث (keyward) هو أى أمر أو دالة من برنامج (+DBase III). ويمكن الضغط على مفتاح (F1) الذى يؤدى نفس العمل مثل كتابة الأمر (HELP). ويؤدى هذا إلى ظهور القائمة الرئيسية لشاشات المساعدة التى يمكن من خلالها الوصول إلى الأمر المطلوب ومتابعة الشرح الخاص به.

٥٠ - الأمـــر (IF)

ويستخدم هذا الأمر فى تنفيذ مجموعة من الأوامر فى حالة تحقق شرط معين. والصورة العامة له كالآتى :

```
IF   <condition>
     <commands>
[ELSE]
     <commands>
ENDIF
```

ويقوم البرنامج باختبار الشرط (condition) بعد (IF) فإذا تحقق يتم تنفيذ الأوامر (Commands) التالية للأمر (ELSE) ، ويلاحظ أن الأمر (ELSE) اختيارى هنا. أى يمكن عدم كتابته وفى هذه الحالة يتم الإنتقال إلى الأوامر التى تلى الأمر (ENDIF) فى حالة عدم تحقق الشرط.

ويمكن استخدام مجموعات متداخلة من الأوامر (IF). وفى هذه الحالة يجب ملاحظة وجود الأمر (ENDIF) الخاص بكل مجموعة داخل المجموعة الخارجية. ويمكن ملاحظة ذلك عن طريق كتابة تعليق (Comment) بعد الأمر (ENDIF) حيث أن هذا التعليق لايؤثر فى تنفيذ البرنامج.

٥١ - الأمـــر (INDEX)

يستخدم هذا الأمر فى إنشاء ملف فهرسى يحتوى على حقلين فقط أحدهما يحتوى على أرقام السجلات الموجودة فى ملف قاعدة البيانات والآخر يحتوى على محتويات الحقل الفهرسى (Index Key) الخاصة بكل سجل. والصورة العامة لهذا الأمر كالآتى :

INDEX ON <index key> TO <filename>[UNIQUE]

حيث :

(Index Key) هو المفتاح الذى يتم الترتيب بناء عليه وهو قد يكون حقلا معينا أو علاقة تشمل عدة حقول. ويجب ملاحظة أن حقول الملاحظات (Memo Fields) لاتستخدم فى هذا المفتاح وكذلك الحقول المنطقية (Logical Fields).

و (Filename) هو إسم الملف الذى يتم إنشاؤه وهو يشمل رمز وحدة الأقراص المستخدمة إذا كان الملف سيتم تخزينه فى وحدة أقراص غير الوحدة الحالية (Current Drive). كما أن الإسم يضاف إليه الإمتداد (NDX.) آليا.

والاختيار (UNIQUE) هنا إختيارى وهو يعنى أنه لاتوجد داخل الفهرس سوى قيمة واحدة لكل سجل. ويعنى ذلك أنه إذا كان هناك أكثر من سجل يحتوى على نفس القيمة للحقل الفهرسى يتم إدخال أول سجل يحتوى على هذه القيمة فى ملف الفهرس.

ويجب ملاحظة أن الفهرسة تكون دائما تصاعدية (Ascendingly) كما أنها لاتغير الترتيب الفعلى للسجلات ولكنها تقوم بترتيب السجلات عند عرضها أو التعامل معها فى الذاكرة المؤقتة. أى أنها تعطى صورة مرتبة للسجلات مع الإحتفاظ بالأصل فى الملف دون أى تغيير.

ويمكن إضافة مجموعة من الحقول فى الحقل الفهرسى. وعند إضافة أنواع مختلفة من الحقول مثل الحقول العددية والحقول التاريخية يتم تحويل كل نوع من هذه الأنواع إلى حقول حرفية. حيث يتم تحويل الحقول العددية عن طريق الدالة (STR) كما يتم تحويل

الحقول التاريخية عن طريق الدالة (DTOC). وأقصى طول للحقل الفهرسى هو (١٠٠) حرف. فمثلا إذا أريد إنشاء ملف فهرسى يحتوى على التاريخ والإسم يتم كتابة السطر التالى :

```
INDEX ON STR(YEAR(<date>),4)+STR(MONTH(<date>),2)+ ;
STR(DAY(<date>),2 ) + Name TO <filename)
```

وفى هذه الحالة يتم ترتيب الملف بناء على التاريخ أولا ثم يتم ترتيب الأسماء داخل كل تاريخ.

٥٢ - الأمـــــر (INPUT)

يستخدم هذا الأمر فى توجيه المستخدم إلى إدخال قيمة معينة وتخزين هذه القيمة فى متغير ذاكرة. والصورة العامة له كالآتى :

```
INPUT  [<message>]  TO  <memvar>
```

حيث :

(message) هى رسالة حرفية تظهر للمستخدم لتوجيهه إلى إدخال المتغير المطلوب. وهى تكون محصورة بين علامات تنصيص فردية 'Single quotes' أو مزدوجة "double quotes" أو أقواس مربعة (Brackets). وقد تستخدم متغيرات الذاكرة مكان هذه الرسالة وفى هذه الحالة لاتوضع بين علامات تنصيص (Quotes).

وعند إدخال المستخدم للقيمة المطلوبة يتم تخزين هذه القيمة فى متغير الذاكرة (memvar) ويتحدد نوع هذا المتغير حسب نوع المدخلات التى يدخلها المستخدم. فمثلا عندما يقوم بإدخال عدد يتم إنشاء متغير عددى لتخزين هذا العدد. ويجب ملاحظة أن القيم الحرفية يتم إدخالها بين علامات تنصيص (Quotes). وهذا الأمر يشبه الأمر (ACCEPT) ولكن الأمر (ACCEPT) يتعامل مع المدخلات الحرفية فقط.

مثـال

لتوجيه المستخدم إلى إدخال إسمه وتخزين هذا الإسم فى متغير الذاكرة (mname)

يتم كتابة السطر التالى :

INPUT "What is your name" TO mname

٥٣ - الأمــــر (INSERT)

يستخدم هذا الأمر فى إضافة سجل جديد إلى ملف قاعدة البيانات عند المكان الذى يقف عنده مؤشر السجلات (Record Pointer) والصورة العامة له كالآتى :

INSERT [BLANK][BEFORE]

ويلاحظ هنا أنه يمكن استخدام الأمر (INSERT) بدون أى اختيارات أخرى وفى هذه الحالة تظهر الحقول الخاصة بسجل جديد يتم إدخال بياناته. وعند الإنتهاء من إدخال البيانات يتم تخزين هذه البيانات بالضغط على مفتاحى (Ctrl-End) وفى هذه الحالة يتم تخزين هذا السجل بعد السجل الحالى (Current Record) الذى يقف عنده المؤشر. وعند استخدام هذا الأمر مع الملف المفهرس (Indexed File) فإنه يعمل مثل الأمر (APPEND) تماما حيث يضيف سجلا فى آخر الملف ثم يضيف سجلات أخرى واحدا تلو الآخر.

والإختيار (BEFORE) يفتح سجلا قبل السجل الحالى. فمثلا إذا كان السجل الحالى هو السجل رقم ٨ فإن الأمر (Insert Before) يفتح سجلا جديدا يكون رقمه (٨) فى حين يصبح السجل السابق رقم (٩).

والاختيار (BLANK) يؤدى إلى إضافة سجل جديد خال مكان السجل الحالى بحيث يمكن إضافة البيانات إليه فيما بعد.

٥٤ - الأمــــر (JOIN)

يستخدم هذا الأمر فى إنشاء ملف جديد عن طريق دمج حقول وسجلات من ملفين مفتوحين. وهو يؤدى عمل الأمر (UPDATE) ولكنه يختلف عنه فى أنه ينشىء ملفا جديدا. أما الأمر (UPDATE) فإنه يعدل ملفا بناء على ملف آخر. والصورة العامة لهذا الأمر كالآتى :

```
JOIN WITH <alias> TO <filename>
    FOR <condition>[FIELDS <fieldlist>]
```

حيث (filename) هو الملف الجديد الذى يتم إنشاؤه ويجب أن يتضمن رمز وحدة الأقراص إذا كان يراد تخزينه على وحدة أقراص غير وحدة الأقراص الحالية (Current Drive).

والإختيار (alias) هو المرادف الذى يشمل إسم الملف المراد دمجه ومنطقة العمل الخاصة بهذا الملف. ويراعى أن يكون الملف المراد دمجه مفتوحا وأن تكون منطقة العمل (Work Area) الخاصة به قد تم اختيارها (Selected) كمنطقة العمل العاملة (Active).

والإختيار (FIELDS <field list>) يستخدم لتحديد أسماء الحقول المراد دمجها من الملفين. وفى حالة عدم تحديد هذه الحقول يتم دمج جميع الحقول فى الملف الجديد.

ملاحظة

عندما يراد تحديد حقول من منطقة عمل غير مختارة (Unselected Work Area) يستخدم التعبير (Alias --> field name) حيث (Alias) هو الإسم المرادف لمنطقة العمل غير المختارة.

٥٥ - الأمـــر (LABEL)

يؤدى هذا الأمر إلى استخدام ملف العناوين المختصرة (Label) الذى سبق إنشاؤه. والصورة العامة له كالآتى :

```
LABEL FORM <filename>/?[<scope>][SAMPLE]
  [WHILE <condition>][FOR <condition>][TO PRINT]
  [TO FILE <flename>]
```

حيث (filename) هو إسم ملف العناوين البريدية المراد استخدامه ويجب أن يشمل رمز وحدة الأقراص إذا كانت غير وحدة الأقراص المستخدمة (Current Drive).

وتستخدم علامة الإستفهام (?) عندما يراد عرض أسماء ملفات العناوين الموجودة على القرص حتى يستطيع المستخدم اختيار الملف المطلوب استخدامه.

والإختيار (Scope) هو المدى المطلوب البحث خلاله عن السجلات المطلوب عرض بياناتها فى التقرير.

والإختيار (SAMPLE) يستخدم لاختبار ضبط التقرير على الطابعة. ويمكن تكرار عملية الإختبار عدة مرات بكتابة (Y) عند ظهور السؤال التالى :

Do you want more samples?

والإختيار (TO FILE) يستخدم عندما يراد تخزين التقرير فى ملف نصوص (Text File) حتى يمكن التعامل معه بواسطة برامج أخرى.

والإختيارات (WHILE) ، (FOR) تستخدم لتحديد السجلات المطلوب عرض بياناتها فى التقرير.

٥٦ - الأمــــر (LIST)

يستخدم هذا الأمر فى عرض بيانات ملف قاعدة البيانات. والصورة العامة له كالآتى :

LIST [OFF][<scope>][<Field list>][WHILE <condition>]
[FOR <condition>][TO PRINT]

ويلاحظ هنا أن جميع الإختيارات مع الأمر إختيارية أى يمكن كتابته دون كتابة أى شىء بعده وفى هذه الحالة يتم عرض بيانات جميع سجلات قاعدة البيانات. وهو فى هذا يختلف عن الأمر (DISPLAY) الذى يعرض بيانات السجل الحالى فقط. كما أن الأمر (LIST) لايوقف الشاشة عند إمتلائها بالبيانات بل يتم الإنتقال إلى الشاشات التالية فورا. وعندما يراد إيقاف الشاشة (Pause) يتم الضغط على مفتاحى (Ctrl-S).

والإختيار (OFF) يستخدم عندما يراد عرض البيانات بدون أرقام السجلات.

والإختيار (scope) يستخدم لتحديد المدى الذى يتم البحث خلاله عن السجلات المراد عرضها.

والإختيار (Field List) يستخدم لتحديد أسماء الحقول المراد عرض بياناتها.

والإختيارات (WHILE) ، (FOR) تستخدم لتحديد السجلات المراد عرض بياناتها.

والإختيار (TO PRINT) يستخدم عندما يراد طباعة البيانات.

٥٧ - الأمــــر (LIST HISTORY)

يستخدم هذا الأمر فى عرض آخر عشرين أمرا سبق تنفيذها وهو يماثل الأمر (DISPLAY HISTORY) ولكنه لايؤدى إلى توقف الشاشة عندما تزيد الأوامر عن طول الشاشة. والصورة العامة له كالآتى :

LIST HISTORY [LAST <N>] [TO PRINT]

والإختيار (<LAST <N>) يستخدم لتحديد عدد الأوامر المطلوب عرضها.

والإختيار (TO PRINT) يستخدم لطباعة الأوامر المعروضة.

٥٨ - الأمــــر (LIST MEMORY)

يستخدم هذا الأمر فى عرض حالة متغيرات الذاكرة (Memory Variables) الموجودة فى الذاكرة المؤقتة (RAM) فى أى وقت ، حيث يعرض إسم كل متغير ونوعه وحجمه ، كما يعرض أيضا بيانات عن حجم الذاكرة المستخدمة وحجم الذاكرة المتاحة. والصورة العامة لهذا الأمر كالآتى :

LIST MEMORY [TO PRINT]

وهو يماثل الأمر (DISPLAY MEMORY) تماما ويختلف عنه فقط فى أنه لايوقف عرض البيانات عندما تزيد عن طول الشاشة.

٥٩ - الأمــــر (LIST STATUS)

يستخدم هذا الأمر فى عرض بيانات عن الحالة الحالية للبرنامج والصورة العامة له كالآتى :

LIST STATUS [TO PRINT]

وهو يؤدى إلى عرض بيانات عن ملفات قاعدة البيانات المفتوحة ورقم منطقة العمل (Work Area) المستخدمة والعلاقات المستخدمة بين الملفات إن وجدت وملفات الفهرس المفتوحة و ... الخ. وهذا الأمر يماثل الأمر (DISPLAY STATUS) تماما ولكنه لايتوقف عند إمتلاء الشاشة بالبيانات.

٦٠ - الأمــــر (LIST STRUCTURE)

يستخدم هذا الأمر فى عرض بيانات عن ملف قاعدة البيانات المفتوح وتركيبه (Structure) الذى يشمل إسم كل حقل ونوعه وعرضه. كما يعرض عدد الحروف (Bytes) التى يتكون منها السجل الواحد والصورة العامة لهذا الأمر كالآتى :

LIST STRUCTURE [TO PRINT]

وهو يماثل الأمر (DISPLAY STRUCTURE) تماما ولكنه لايتوقف عند إمتلاء الشاشة بالبيانات.

٦١ - الأمــــر (LOAD)

يستخدم هذا الأمر فى كتابة برامج فرعية بلغة التجميع (Assembly) وتشغيل هذه البرامج من خلال البرنامج الذى يتم تصميمه بواسطة (+DBase III). ويجب أن تكون هذه البرامج على هيئة الشفرة الثنائية (Binary Code). والصورة العامة لهذا الأمر كالآتى :

LOAD <filename>

وهذا يؤدى إلى تحميل الملف (Filename) فى الذاكرة المؤقتة حتى يتم استدعاؤه فى

البرنامج بواسطة الأمر (CALL). ويسمح برنامج (+DBase III) بتحميل حتى خمسة برامج من هذا النوع فى كل مرة وكل برنامج بحد أقصى ٣٢ ك بايت بشرط أن تكون بالشفرة الثنائية (Binary).

٦٢ - الأمـــر (LOCATE)

يستخدم هذا الأمر للبحث فى ملف قاعدة البيانات عن السجل الذى يحقق شرطا أو شروطا معينة. والصورة العامة له كالآتى :

```
LOCATE [<scope>] [WHILE <condition>]
    [FOR <condition>]
```

والإختيار (scope) يستخدم لتحديد مدى معين من السجلات للبحث خلاله. فمثلا عند كتابة (NEXT <N>) فإن البحث يتم خلال العدد (N) من السجلات التى تلى السجل الحالى (Current Record).

والإختيارات (WHILE) ، (FOR) تستخدم لتحديد السجلات المطلوب البحث عنها. ويجب ملاحظة أن هذا الأمر يؤدى إلى توجيه المؤشر إلى أول سجل يحقق الشروط. فإذا أريد تحريكه إلى باقى السجلات يستخدم الأمر (CONTINUE).

٦٣ - الأمـــر (LOOP)

يستخدم هذا الأمر فى الإنتقال إلى بداية الحلقة التكرارية دون تنفيذ باقى أوامر الحلقة. والصورة العامة له كالآتى :

```
LOOP
```

ويستخدم فى معظم الأحيان فى البرامج الفرعية (Modules) الخاصة بقوائم الإختيارات (Menus) فمثلا يمكن كتابة الأوامر التالية فى قاعدة بيانات الطلبة (Cadets) :

```
USE Cadets
mchoice = " "
```

```
DO WHILE mchoice =" "
  CLEAR
  @ 5,5 SAY "A-Add records"
  @ 7,5 SAY "E-Edit records"
  @ 9,5 SAY "R-Return to dot prompt"
  READ
  IF .NOT. mchoice $ "AaEeRr"
        LOOP
  ENDIF
  DO CASE
        CASE mchoice $ "Aa"
              APPEND
        CASE mchoice $ "Ee"
              EDIT
        CASE mchoice $ "Rr"
              EXIT
  ENDCASE
ENDDO
CLOSE DATABASES
```

ويتم اختبار المتغير (mchoice) بعد الأمر (IF) فإذا لم يكن الحرف المخزن به ضمن الحروف (AaEeRr) يتم تنفيذ الأمر التالى وهو (LOOP). وفى هذه الحالة يتم الإنتقال إلى أول الحلقة التكرارية مرة ثانية لاختبار الحرف الجديد الذى يدخله المستخدم فى المتغير (mchoice). والدالة ($) تعنى هنا البحث عن الحرف الموجود فى المتغير (mchoice) خلال السلسلة الحرفية (AaEeRr). وفى هذه السلسلة تم وضع الحروف الممثلة لاختيارات القائمة سواء كانت صغيرة (Small) أو كبيرة (Capital).

٦٤ - الأمـــــر (MODIFY COMMAND)

يستخدم هذا الأمر فى إنشاء وتعديل ملفات الأوامر (Command files) التى تستخدم فى كتابة البرامج الخاصة ببرنامج (+DBase III). وهذا الأمر يؤدى إلى تشغيل برنامج تنسيق النصوص (Text editor) الخاص ببرنامج (+DBase III) والذى يمكن

استخدامه فى كتابة أى نص (Text) والصورة العامة للأمر كالآتى :

MODIFY COMMAND <filename>

ويمكن أن يحتوى إسم الملف (filename) على رمز وحدة الأقراص المراد تخزين الملف عليها إذا لم تكن هى وحدة الأقراص الحالية (Current Drive). وهذا الأمر يضيف الإمتداد (prg.) إلى إسم الملف آليا. أما إذا أريد إضافة أى إمتداد آخر فيجب فى هذه الحالة إضافة الإمتداد إلى إسم الملف المراد إنشاؤه. وعند كتابة هذا الأمر فإن البرنامج يبحث عن ملف بالإسم (Filename) فإذا وجده فإنه يستدعيه ويعرض الملف لتصحيحه وإذا لم يجده فإنه ينشىء ملفا جديدا بهذا الإسم. وهذا الأمر يسمح بإنشاء ملف حجمه حتى ٥٠٠٠ حرف (Byte). أما اذا استخدم برنامج تنسيق نصوص خارجى من خلال ملف المواصفات (Configuration File) فإنه يمكن فى هذه الحالة زيادة عدد الحروف عن ذلك.

وعند كتابة أمر يزيد عن طول السطر يمكن كتابة الفاصلة المنقوطة (;) فى نهاية السطر ثم استكمال كتابة الأمر فى السطر التالى. وعندما يراد طباعة محتويات الملف يستخدم الأمر (TYPE <filename> TO PRINT) من مشيرة النقطة (Dot Prompt).

٦٥ - الأمـــر (MODIFY LABEL)

إرجع إلى (CREATE/MODIFY LABEL).

٦٦ - الأمـــر (MODIFY QUERY)

إرجع إلى (CREATE/MODIFY QUERY).

٦٧ - الأمـــر (MODIFY REPORT)

إرجع إلى (CREATE/MODIFY REPORT).

٦٨ - الأمـــر (MODIFY SCREEN)

إرجع إلى (CREATE/MODIFY SCREEN).

٦٩ - الأمـر (MODIFY STRUCTURE)

ويستخدم هذا الأمر عندما يراد تعديل تركيب ملف قاعدة البيانات والصورة العامة له كالآتى :

MODIFY STRUCTURE <Filename>

وهذا الأمر ينشىء دائما ملفا إحتياطيا بالإمتداد (bak.) لكل ملف يتم تعديله. وعند الإنتهاء من إدخال التعديلات يقوم بإضافة السجلات (Append) إلى الملف الأصلى. وإذا تم تعديل أى حقول ملاحظات يتم إنشاء ملف إحتياطى للملف الملاحظات بالإمتداد (&&&.).

٧٠ - الأمــــر (NOTE/*)

يستخدم الأمر (NOTE) أو النجمة (*) لنفس الهدف وهو كتابة ملاحظات لتوضيح وظائف أوامر البرنامج المختلفة. ويتم كتابة هذا الأمر فى بداية السطر المطلوب كتابة ملاحظات فيه والصورة العامة لهذا الأمر كالآتى :

NOTE/* <text>

وعندما يراد كتابة عدة سطور ملاحظات متتالية يمكن كتابة فاصلة منقوطة (;) فى نهاية كل سطر. ويمكن كتابة الملاحظات فى أى مكان فى السطر باستخدام الحرفين (&&).

مـثال

هذا المثال يوضح استخدام الأمر (NOTE) فى برنامج.

```
NOTE This is a loop
STORE 1 TO X
DO WHILE X < 100
   STORE X + 1 TO X
ENDDO
```

كما يمكن استخدام الحرفين (&&) كالآتى مثلا :

mname = SPACE (35) && initiate a memory variable

٧١ - الأمـــر (ON)

يستخدم هذا الأمر فى تنفيذ أمر أو برنامج فرعى بناء على تحقيق شرط معين والصورة العامة له كالآتى :

ON ERROR/ESCAPE/KEY <command>

وهناك ثلاثة صور للأمر وهى كالآتى :

ON ERROR <command>	وهو يؤدى إلى تنفيذ أمر معين أو مجموعة من الأوامر فى حالة ظهور خطأ (Error) فى البرنامج.
ON ESCAPE <command>	وهو يؤدى إلى تنفيذ أمر معين أو مجموعة من الأوامر فى حالة ضغط المستخدم على مفتاح الهروب (ESC).
ON KEY <command>	وهو يؤدى إلى تنفيذ أمر معين أو مجموعة من الأوامر فى حالة ضغط المستخدم على أى مفتاح.

ويظل هذا الأمر مؤثرا فى البرنامج حتى إنهائه بواسطة الأمر (ON ERROR/ESCAPE/KEY) وذلك دون كتابة أى شىء بعده. ويجب ملاحظة أن الأمر (ON ESCAPE) لايعمل مع الأمر (SET ESCAPE OFF). كما يجب ملاحظة أن الأمر (ON ESCAPE) له أولوية على الأمر (ON KEY) وهذا يعنى أنه عند كتابة الأمرين معا فى نفس البرنامج فإن الضغط على مفتاح الهروب يؤدى إلى تنفيذ الأمر أو الأوامر التى تلى الأمر (ON ESCAPE) وليس الأوامر التى تلى الأمر (ON KEY). والأمر (ON ERROR) يتعامل فقط مع الأخطاء المتعلقة بقواعد أوامر (+DBase III) وهى التى تسمى أخطاء القواعد (Syntax Errors).

مثال

يمكن تشغيل البرنامج (error1) عند ظهور أى خطأ فى البرنامج وذلك عن طريق كتابة السطر التالى :

```
ON ERROR DO error1
```

كما يمكن كتابة الأمر التالى :

```
ON ESCAPE EXIT
```

وذلك للخروج من حلقة تكرارية مثلا فى حالة ضغط المستخدم على مفتاح الهروب (ESC). كما يمكن إيقاف تأثير مفتاح الهروب عن طريق كتابة الأمر (ON ESCAPE) دون كتابة أى شيء بعده.

٧٢ - الأمـــر (PACK)

يستخدم هذا الأمر فى مسح السجلات التى سبق وضع علامات عليها تمهيدا لمسحها من الملف والصورة العامة له كالآتى :

```
PACK
```

٧٣ - الأمـــر (PARAMETERS)

يستخدم هذا الأمر فى تخصيص معاملات فى البرنامج الفرعى يتم من خلالها استقبال القيم التى يتم إدخالها عند استدعاء هذا البرنامج الفرعى. وعند استدعاء هذا البرنامج الفرعى يستخدم الأمر (DO <filename> WITH <parameter list>) حيث يتم إدخال القيم المقابلة للمتغيرات التى سبق تخصيصها والصورة العامة لهذا الأمر كالآتى :

```
PARAMETERS <parameter list>
```

ويكتب هذا الأمر فى بداية البرنامج الفرعى سواء كان برنامج خطوات (Procedure) أو أى برنامج فرعى آخر. وقائمة المعاملات (Parameter list) هى المتغيرات التى

تستقبل المعاملات التى يتم إمرارها مع الأمر (DO). ويجب أن يكون عدد المعاملات فى أمر الإستقبال (PARAMETERS) وأمر الإمرار (DO) واحدا. وإذا كان المعامل الذى يتم إدخاله أو إمراره (Passing) متغير ذاكرة فإن قيمة هذا المتغير قد تتغير بناء على تشغيل البرنامج الفرعى ثم يتم نقل هذه القيمة إلى البرنامج الذى قام باستدعاء البرنامج الفرعى.

مثـــال

الأوامر التالية تمثل برنامج خطوات أو إجراءات (Procedure) إسمه (Calc) :

```
PARAMETERS length , width , area
area = length * width
RETURN
```

ولتنفيذ هذا البرنامج مع إمرار القيمة (7) للطول (Length) والقيمة (9) للعرض (width) يتم كتابة الأمر التالى :

```
DO Calc WITH 7, 9, area
```

وللسؤال عن القيمة المخزنة فى المتغير (area) يتم كتابة الأمر التالى :

```
? area
```

وفى هذه الحالة يظهر العدد (63) الذى يمثل نتيجة المعادلة الموجودة فى البرنامج (Calc).

٧٤ - الأمـــر (PRIVATE)

يستخدم هذا الأمر عندما يراد استخدام متغيرات ذاكرة فى برنامج فرعى بحيث لاتتأثر بأى متغيرات أخرى تحمل نفس أسماء هذه المتغيرات وتكون موجودة فى البرنامج الرئيسى أو فى أى برامج أخرى غير البرامج التابعة لهذا البرنامج الفرعى. كما يستخدم هذا الأمر أيضا لتغيير حالة متغير الذاكرة من متغير عام (Public) إلى متغير خاص (Private). وهناك صورتان للأمر كالآتى :

PRIVATE <memvar list>
PRIVATE ALL [LIKE/EXCEPT <skeleton>]

ويستخدم الإختيار (LIKE) عندما يراد استخدام الحروف الشاملة فى تغيير عدد من المتغيرات العامة (Public) إلى خاصة (Private). كما يستخدم الإختيار (EXCEPT) عندما يراد تحويل كل المتغيرات العامة إلى متغيرات خاصة ماعدا بعض المتغيرات التى يتم تحديدها عن طريق استخدام الحروف الشاملة (Global Characters) فمثلا لتحويل كل المتغيرات التى تبدأ بالحرف (m) إلى متغيرات خاصة يستخدم الأمر التالى :

PRIVATE ALL LIKE m*

كما يمكن تحويل كل المتغيرات إلى متغيرات خاصة ماعدا المتغيرات التى تبدأ بالحرف (s) كالآتى :

PRIVATE ALL EXCEPT s*

ويجب ملاحظة أن المتغيرات الخاصة تختفى القيم الموجودة بها بمجرد إنتهاء البرنامج وذلك عكس المتغيرات العامة (Public) التى تظل محتفظة بآخر قيمة تم إدخالها فيها.

٧٥ - الأمـــر (PROCEDURE)

يستخدم هذا الأمر لتحديد بداية كل برنامج خطوات أو إجراءات (Procedure) داخل ملف الخطوات (Procedure file) والصورة العامة له كالآتى :

PROCEDURE <procedure name>

وكل برنامج خطوات أو إجراءات (Procedure) يبدأ بالأمر (PROCEDURE) يليه إسم برنامج الخطوات وإسم برنامج الخطوات يتكون من ٨ حروف. ويمكن أن يحتوى على حروف أو أرقام أو شرطة سفلية (Underscore) كما يجب أن يبدأ بحرف (Character).

مثـال

يمكن أن يحتوى ملف الخطوات أو الإجراءات (Proc1) على برامج الخطوات الآتية :

```
PROCEDURE message
@ 10,0 CLEAR
@ 15,0 SAY "This is an invalid data"
RETURN

PROCEDURE printer
@ 10,0 CLEAR
 @ 15,0 SAY "Send reprot to printer? Y/N" GET Pr ;
    PICTURE"!"
 IF Pr = Y
         SET PRINT ON
 ENDIF
 RETURN
```

ولتشغيل برنامج الخطوات (Printer) مثلا يتم أولا فتح ملف الخطوات (Proc1) وذلك كالآتى :

```
SET PROCEDURE TO Proc1
DO Printer
```

٧٦ - الأمـــر (PUBLIC)

يستخدم هذا الأمر فى تحويل المتغيرات إلى متغيرات عامة يمكن استخدامها فى أى برنامج فرعى أو رئيسى. وهذه المتغيرات تختلف عن باقى متغيرات الذاكرة فى أن القيم المخزنة بها تظل موجودة حتى بعد إنتهاء البرنامج وتظل موجودة فى الذاكرة المؤقتة حتى يتم مسحها باستخدام الأمر (RELEASE) أو (CLEAR MEMORY) والصورة العامة للأمر كالآتى :

```
PUBLIC <memory variable list>
```

وعند استخدام هذا الأمر فإن أى برنامج فرعى أو رئيسى يستخدم هذه المتغيرات يمكنه تغيير القيم الموجودة فيها. ويجب ملاحظة أن المتغيرات العامة يجب إعلانها عامة (Declaring) قبل إعطائها أى قيمة فمثلا الأوامر التالية تعتبر خطأ :

```
Answer = "Y"
PUBLIC Answer
```

وذلك لأن المتغير (Answer) قد تم إعطاؤه القيمة "Y" قبل إعلانه عاما.

٧٧ - الأمـــر (QUIT)

يستخدم هذا الأمر فى إغلاق جميع الملفات والخروج من البرنامج والرجوع إلى نظام التشغيل والصورة العامة له كالآتى :

```
QUIT
```

ويساعد هذا الأمر على الخروج من البرنامج بطريقة آمنة دون تحطيم أى ملفات حيث أن إغلاق جهاز الحاسب دون استخدام هذا الأمر قد يؤدى إلى تحطيم الملفات المفتوحة وفقد البيانات المخزنة فيها.

٧٨ - الأمـــر (READ)

يستخدم هذا الأمر فى تخزين البيانات التى يتم إدخالها عن طريق الأمر (@...GET). وهو يستخدم عادة فى تصميم شاشات الإدخال من خلال البرنامج والصورة العامة له كالآتى :

```
READ [SAVE]
```

والإختيار (SAVE) يستخدم للإبقاء على البيانات مخزنة فى الـ (GETS) حتى يتم مسحها بواسطة الأمر (CLEAR GETS) حيث أن الوضع الطبيعى أن يتم مسح هذه الـ (GETS) بمجرد إدخال البيانات فى متغيرات الذاكرة فمثلا عند كتابة الأوامر التالية :

```
STORE "  " TO mname
@ 10,10 SAY "Enter your name:" GET mname
READ
```

فى هذه الحالة يتم تخزين الإسم الذى يدخله المستخدم فى مخزن مؤقت (GET). وعندما يقابل البرنامج الأمر (READ) فإنه ينقل هذا الإسم إلى متغير الذاكرة (mname) وبالتالى يتم إخلاء المخزن المؤقت (GET).

أما عند استخدام الأمر (READ SAVE) بدلا من الأمر (READ) فى المثال السابق فإن المخزن المؤقت (GET) يظل محتفظا بالإسم الذى تم إدخاله ولذلك يراعى عند استخدام الأمر (READ SAVE) عدة مرات التأكد من مسح المخازن المؤقتة (GETS) قبل أن يصل عددها إلى الحد الأقصى المسموح به فى البرنامج وهو (١٢٨) مخزنا فى الوضع المبدئى (Default).

ويستخدم الأمر (READ) أيضا عند تصميم شاشات الإدخال التى تتضمن أكثر من شاشة حيث يتم كتابة الأمر (READ) فى نهاية كل شاشة ويصبح فى هذه الحالة ناقل للصفحة التالية (Page Break). ويستطيع المستخدم فى هذه الحالة الإنتقال من أى شاشة إلى الشاشة الأخرى باستخدام مفتاحى (PgDn) (PgUp).

٧٩ - الأمـــر (RECALL)

يستخدم هذا الأمر فى استعادة السجلات التى سبق وضع علامات عليها تمهيدا لمسحها باستخدام الأمر (PACK) والصورة العامة له كالآتى :

```
RECALL [<scope>][WHILE <condition>]
        [FOR <condition>]
```

والإختيارات كلها إختيارية فى هذه الحالة حيث يمكن كتابة الأمر دون كتابة أى شىء بعده وفى هذه الحالة يتم استعادة أول سجل فقط من السجلات التى سبق وضع علامات عليها لمسحها. وتستخدم الإختيارات فى إستعادة السجلات التى تحقق شروطا معينة.

فمثلا لاستعادة أول سجل تم تجهيزه للمسح يتم كتابة السطر التالى :

RECALL

وإذا أريد استعادة السجل رقم (١٠) يتم كتابة السطر التالى :

RECALL RECORD 10

وإذا أريد إستعادة جميع السجلات التى تم تجهيزها للمسح يتم كتابة السطر التالى :

RECALL ALL

حيث تمثل (ALL) المدى (scope) الذى يشمل كل السجلات التى تم تجهيزها للمسح.

٨٠ - الأمـــر (REINDEX)

يستخدم هذا الأمر فى إعادة إنشاء ملف الفهرس المفتوح ، وذلك حتى يتم تحديث هذا الملف فى بعض الحالات التى تتطلب ذلك. والصورة العامة له كالآتى :

REINDEX

ويجب مراعاة فتح ملف الفهرس المراد تحديثه قبل كتابة هذا الأمر.

٨١ - الأمـــر (RELEASE)

يستخدم هذا الأمر فى مسح متغيرات الذاكرة (Memory Variables) مما يتيح استخدام الذاكرة المتاحة فى إدخال متغيرات أخرى والصورة العامة له كالآتى :

RELEASE <memvar list> [ALL]
[LIKE/EXCEPT <skeleton>] [MODULE <module name>]

ويستخدم الإختيار (skeleton) عندما يراد استخدام الحروف الشاملة (Global) فى تحديد المتغيرات المطلوب مسحها أو إستثناء بعض المتغيرات من المسح حسب الحاجة.

كما يستخدم الإختيار (LIKE) لتحديد المتغيرات المشتركة فى حرف معين أو عدة حروف.

ويستخدم الإختيار (EXCEPT) فى إستثناء المتغيرات المشتركة فى حرف معين أو عدة حروف.

كما يستخدم الإختيار (ALL) فى مسح جميع متغيرات الذاكرة.

ويستخدم الإختيار (MODULE) فى مسح برنامج مكتوب بلغة التجميع (Assembly) من الذاكرة ويساعد هذا على استخدام عدة برامج خارجية تزيد عن الحد الأقصى لعدد هذه البرامج وهو خمسة.

أمثلة

لمسح كل متغيرات الذاكرة التى تبدأ بالحرف (m) يتم كتابة السطر التالى :

RELEASE ALL LIKE m *

ولمسح جميع متغيرات الذاكرة ماعدا المتغيرات التى تبدأ بحرف (s) يتم كتابة السطر التالى :

RELEASE ALL EXCEPT s*

كما يمكن مسح متغيرات ذاكرة معينة بكتابة السطر التالى مثلا :

RELEASE mname , maddress , mage

٨٢ - الأمـــر (RENAME)

يستخدم هذا الأمر فى تغيير إسم ملف سبق تخزينه على القرص إلى إسم جديد والصورة العامة لهذا الأمر كالآتى :

```
RENAME <old filename> TO <new filename>
```

ويجب أن يتضمن الإسم القديم والإسم الجديد الإمتداد ورمز وحدة الأقراص المخزن عليها الملف إذا كانت غير وحدة الأقراص الحالية (Current Drive). وفى حالة تغيير إسم ملف قاعدة بيانات يحتوى على حقول ملاحظات (memo fields) يتم تغيير إسم ملف الملاحظات منفصلا.

مثال

لتغيير إسم الملف (Cadets) إلى الإسم (Grades) يتم كتابة السطر التالى :

```
RENAME Cadets.dbf TO Grades.dbf
```

٨٣ - الأمـــر (REPLACE)

يستخدم هذا الأمر فى استبدال محتويات حقول معينة بقيم جديدة يتم إدخالها ويتم الحصول على هذه القيم عادة من متغيرات ذاكرة (Memory Variables) والصورة العامة لهذا الأمر كالآتى :

```
REPLACE[<scope>]<field1>
  WITH <exp>[,<field2> WITH <exp>]
  [WHILE <condition>][FOR <condition>]
```

وفى حالة عدم إدخال مدى (scope) أو شروط معينة للبحث فإن الإستبدال يتم على السجل الحالى (Current Record). كما يتم تحديد الحقول التى يتم إستبدالها عن طريق كتابة إسم كل حقل مكان الاختيارات (field1, field2, ...etc). ويمكن إستبدال جميع حقول السجل بقيم جديدة.

وفى حالة إستبدال الحقل الفهرسى (Key Field) بقيم جديدة فإن ذلك سوف يؤدى إلى تحديث الفهرس (Updating) وبالتالى يحدث تغير فى ترتيب السجلات لذلك يراعى عند استبدال الحقل الفهرسى عدم استخدام المدى (scope) وأوامر البحث مثل (WHILE , FOR) لأن ترتيب السجلات يكون قد تغير.

مثـال

لزيادة سعر الوحدة (Unit Cost) بمقدار (١٠ ٪) فى جميع السجلات يتم كتابة السطر التالى :

```
REPLACE ALL Unitcost WITH Unitcost*1.1
```

٨٤ - الأمـــر (REPORT)

يستخدم هذا الأمر لعرض التقرير الذى سبق تصميمه متضمنا بيانات مجموعة من السجلات يتم تحديدها وقد يتم عرض التقرير على الشاشة أو طباعته على الطابعة أو تخزينه كملف آسكى (ASCII file) والصورة العامة لهذا الأمر كالآتى :

```
REPORT FORM <filename>/?[<scope>]
[WHILE <condition>][FOR <condition>]
[PLAIN][HEADING <exp>][NOEJECT]
[TO PRINT/TO FILE <filename>][SUMMARY]
```

ولايلزم هنا كتابة الإمتداد فى إسم الملف (filename) حيث أن البرنامج يضيف الإمتداد (.fmt) آليا. ويمكن تلخيص الإختيارات المختلفة كالآتى :-

الإختيارات (scope) ، (WHILE) ، (FOR) تستخدم لتحديد مدى وشروط البحث التى يتم عن طريقها تحديد السجلات المطلوب عرض بياناتها.

والإختيار (PLAIN) يؤدى إلى عرض تقرير عادى لايحتوى على أرقام الصفحات أو تاريخ اليوم الحالى.

والإختيار (HEADING) يستخدم فى إضافة سلسلة حرفية تمثل عنوانا معينا للتقرير يكتب فى كل صفحة.

والإختيار (NOEJECT) يستخدم لإلغاء نقل الصفحة (Page Break) وهذا يتيح لمخطط البرامج التحكم فى مكان الإنتقال إلى الصفحة التالية وبالتالى يمكنه التحكم فى

طول الصفحة (Page Length).

الإختيار (TO PRINT) يستخدم عندما يراد طباعة التقرير.

الإختيار (TO FILE) يستخدم لتخزين التقرير كملف آسكى (ASCII file) وهذا يتيح التعامل معه من خلال برامج أخرى مثل برامج الجداول الإلكترونية.

الإختيار (SUMMARY) يستخدم فى الحصول على تقرير مختصر يحتوى فقط على تجميع البيانات العددية الموجودة فى الملف وفى حالة استخدام الإختيار (Groups) عند إنشاء التقرير من خلال برنامج المساعد (Assistant) يتم تجميع البيانات الخاصة بكل مجموعة منفصلة. ويراعى فى هذه الحالة فهرسة الملف باستخدام الحقل المستخدم فى التجميع (Grouping) كحقل فهرسى (Key field).

٨٥ - الأمــــر (RESTORE)

يستخدم هذا الأمر فى إسترجاع متغيرات الذاكرة التى سبق تخزينها فى ملف ذاكرة (Memory file). والصورة العامة له كالآتى :

RESTORE FROM <filename> [ADDITIVE]

حيث (filename) هو إسم ملف الذاكرة المطلوب استرجاعه.

والإختيار (ADDITIVE) يستخدم لإضافة المتغيرات الموجودة فى الملف إلى المتغيرات الموجودة فى الذاكرة المؤقتة فى هذا الوقت حيث أن عدم استخدام هذا الإختيار يؤدى إلى مسح جميع المتغيرات الموجودة فى الذاكرة لتحل محلها المتغيرات الموجودة فى الملف.

ويجب ملاحظة أن جميع المتغيرات التى يتم إسترجاعها تصبح خاصة (Private) بصرف النظر عن حالتها داخل الملف. وعند استرجاع الملف من مشيرة النقطة (Dot Prompt) تصبح المتغيرات عامة (Public).

٨٦ - الأمــــر (RESUME)

يستخدم هذا الأمر لإستكمال تنفيذ البرنامج بعد إيقافه بواسطة الأمر(SUSPEND). والصورة العامة له كالآتى :

RESUME

ويؤدى هذا الأمر إلى إستكمال تنفيذ البرنامج من نفس المكان الذى توقف عنده عند استخدام الأمر (SUSPEND). ويستخدم هذا عادة فى اختبار وتصحيح البرنامج (Testing and Debugging).

٨٧ - الأمـــر (RETURN)

يستخدم هذا الأمر فى الرجوع من البرنامج الفرعى إلى البرنامج الذى قام باستدعائه أو الرجوع إلى مشيرة النقطة (Dot Prompt) فى حالة إستدعائه من البرنامج الرئيسى. والصورة العامة له كالآتى :

RETURN [TO MASTER]

والإختيار (TO MASTER) يستخدم للرجوع إلى البرنامج الرئيسى مباشرة من أى برنامج فرعى. وعند الرجوع إلى البرنامج القائم باستدعاء البرنامج الفرعى يتم استكمال تنفيذ الأوامر التى تلى الأمر الذى قام باستدعائه.

واستخدام الأمر (RETURN) فى نهاية البرنامج يؤدى إلى إغلاق هذا البرنامج كما يؤدى إلى مسح متغيرات الذاكرة الخاصة (Private) ولكنه لايؤثر فى المتغيرات العامة (Public).

٨٨ - الأمـــر (RUN/!)

يستخدم هذا الأمر فى تشغيل أمر من أوامر نظام التشغيل (MS-DOS) من خلال برنامج (+DBase III). والصورة العامة له كالآتى :

RUN <command>

كما يمكن كتابته كالآتى :

! <command>

حيث يعمل الحرف (!) نفس عمل الأمر (RUN). ويمكن من خلال هذا الأمر تشغيل أوامر نظام التشغيل مثل (DIR, RENAME , ERASE). ويجب ملاحظة أن هذا الأمر يتطلب ذاكرة مؤقتة أكبر من (٢٥٦) حرفا.

٨٩ - الأمـــــر (SAVE)

يستخدم هذا الأمر فى تخزين متغيرات الذاكرة الموجودة فى الذاكرة المؤقتة فى ملف ذاكرة (Memory File) والصورة العامة له كالآتى :

```
SAVE TO <filename>
        [ALL LIKE/EXCEPT <skeleton>]
```

حيث (filename) هو إسم الملف المطلوب تخزين متغيرات الذاكرة فيه. ويجب أن يحتوى إسم الملف على رمز وحدة الأقراص المراد تخزين الملف فيها إذا كانت غير وحدة الأقراص الحالية (Current Drive). ولايتم كتابة الإمتداد حيث أن البرنامج يضيف الإمتداد (.mem) آليا.

والإختيار (ALL LIKE) يستخدم عندما يراد استخدام الحروف الشاملة فى تحديد مجموعة معينة من متغيرات الذاكرة التى تحتوى أسماؤها على حروف معينة.

والإختيار (ALL EXCEPT) يستخدم عندما يراد إستثناء بعض المتغيرات التى تحتوى على حروف معينة من التخزين فى الملف.

والإختيار (skeleton) يستخدم لإدخال الحروف الشاملة (Global Characters) المطلوب استخدامها.

وهناك حرفان يستخدمان كحروف شاملة وهما الحرف (*) وهو يحل محل أى عدد من الحروف والحرف (?) ويحل محل حرف واحد فقط.

أمثـلة

عندما يراد تخزين جميع متغيرات الذاكرة الموجودة فى الذاكرة المؤقتة فى ملف إسمه

(File1) مثلا يتم كتابة السطر التالى :

SAVE TO B: file1

ولتخزين جميع متغيرات الذاكرة الموجودة فى الذاكرة المؤقتة والتى تبدأ بالحرف (m) فى الملف (File2) الموجود على وحدة الأقراص (B) يتم كتابة السطر التالى :

SAVE ALL LIKE m* TO B: file2

ويجب ملاحظة أنه عند تخزين متغيرات ذاكرة فى ملف سبق إنشاؤه تظهر الرسالة التالية :

file2.mem already exists, overwrite it? (Y/N)

وعند كتابة (Y) فإن المتغيرات الجديدة تمسح أى متغيرات أخرى سبق تخزينها فى الملف. ولتخزين جميع المتغيرات التى لاتحتوى على الحرف (A) كحرف ثان فى إسم المتغير يتم كتابة السطر التالى :

SAVE ALL EXCEPT ?A*

حيث يمثل الحرف الشامل (?) الحرف الأول من إسم المتغير بينما يمثل الحرف الشامل (*) الحروف الباقية من إسم متغير الذاكرة.

٩٠ - الأمـــر (SEEK)

يستخدم هذا الأمر فى البحث خلال ملف قاعدة البيانات المفهرس (Indexed) عن السجل الذى تطابق محتويات الحقل الفهرسى له قيمة معينة يتم إدخالها والصورة العامة له كالآتى :

SEEK <expression >

والتعبير (expression) قد يكون قيمة عددية أو حرفية أو تاريخية أو علاقة حسابية. وإذا كان قيمة حرفية (String) فيجب كتابتها بين علامات تنصيص

(Quotation). وهو فى هذا يختلف عن الأمر الآخر (FIND) الذى لايحتاج إلى وضع القيمة الحرفية المطلوب البحث عنها بين علامات تنصيص.

ويجب ملاحظة أن مقارنة القيمة الحرفية (String) تبدأ من أول حرف وحتى نهاية السلسلة (String). فإذا كانت حروف السلسلة مطابقة لأول حروف فى الحقل الفهرسى فإن الحقل يعتبر مطابقا بصرف النظر عن باقى حروفه وفى هذه الحالة يقف مؤشر السجلات (Record Pointer) عند أول سجل مطابق. وإذا لم يتطابق الحقل مع القيمة الحرفية المطلوبة تظهر الرسالة "No found" وفى هذه الحالة ينتقل المؤشر إلى نهاية الملف.

وللبحث عن تاريخ معين يجب أولا تحويله من حروف (Characters) إلى تاريخ (Date) وذلك باستخدام الدالة (CTOD) كالآتى مثلا :

SEEK CTOD ('01/01/90')

٩١ - الأمــــر (SELECT)

يستخدم هذا الأمر فى التعامل مع مناطق العمل المختلفة (Work Areas) حيث يمكن عن طريق هذا الأمر فتح حتى عشر مناطق عمل وهذا يتيح التعامل مع عشرة ملفات قواعد بيانات فى نفس الوقت. والصورة العامة لهذا الأمر كالآتى :

SELECT <work area/alias>

ومناطق العمل (Work Areas) تأخذ الأرقام من (١) إلى (١٠) أو الحروف من (A) إلى (J). كما يمكن استخدام المرادفات (Aliases) فى تسمية مناطق العمل والملفات الموجودة فيها. ويراعى عند اختيار المرادفات ألا تكون الحروف من(A) إلى (J) وذلك لأن هذه الحروف تمثل المرادفات المبدئية (Default) لمناطق العمل من (١) إلى (١٠). ولكن يمكن الجمع بين الحروف لتكوين مرادفات مثل (AA) أو (BB) وهكذا. ويمكن فتح عدة ملفات قواعد بيانات فى مناطق عمل مختلفة ولكن لايتم التعامل إلا مع آخر منطقة عمل تم فتحها باستخدام الأمر (SELECT).

ويمكن عرض بيانات من مناطق عمل أخرى غير منطقة العمل الحالية (Current Work Area) وذلك باستخدام العلامة (->). وهذه العلامة تتكون من حرف الناقص (-) يليه حرف أكبر من (>). فعند كتابة هذه العلامة بعد إسم منطقة العمل أو

المرادف (Alias) يتم الحصول على البيانات الموجودة فى الحقل الخاص بالسجل الذى يقف عنده المؤشر (Pointer) وذلك كالآتى :

```
Alias -> fieldname
```

ويتم ذلك عادة بعد ربط الملفات الموجودة فى مناطق عمل مختلفة باستخدام الأمر (SET RELATION). ويجب ملاحظة أن كل منطقة عمل لها مؤشر سجلات (Record Pointer) خاص بها كما أن مؤشر السجلات الوحيد الذى يمكن التحكم فيه هو مؤشر منطقة العمل الحالية (Current Work Area) أما باقى المؤشرات فتظل ساكنة. وذلك باستثناء استخدام الأمر (SET RELATION) فإنه يؤدى إلى ربط جميع مؤشرات مناطق العمل المفتوحة بالمؤشر الموجودة فى منطقة العمل الحالية.

أمثلة

عندما يراد ربط ملف الطلبة (Cadets1) بملف الطلبة (Cadets2) يتم فتح كل ملف فى منطقة عمل مختلفة كالآتى :

```
SELECT 1
USE Cadets1
SELECT 2
USE Cadets2 INDEX Name
SELECT 1
SET RELATION TO name INTO cadets2
```

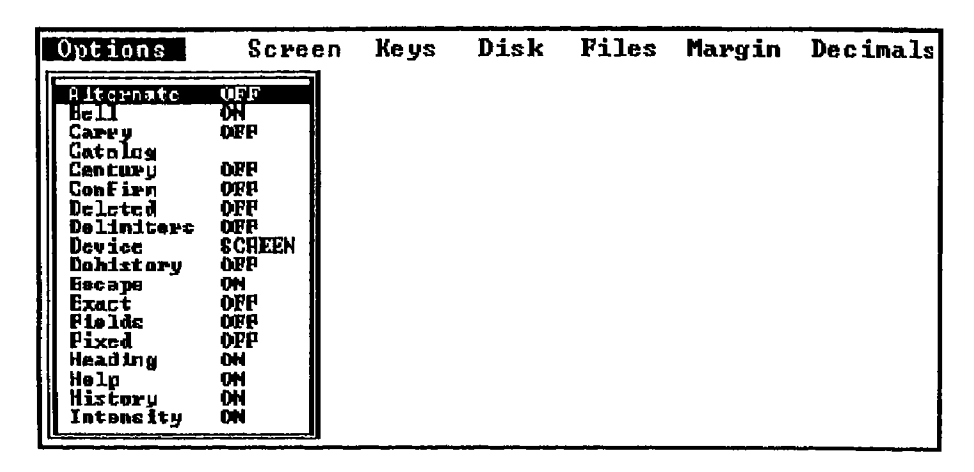
ويلاحظ فى هذه الحالة أن منطقة العمل الحالية (Current) هى المنطقة (١). كما أن الملف الذى يتم ربطه بالملف الموجود فى منطقة العمل الحالية يجب أن يكون مفهرسا بناء على حقل مشترك بين الملفين.

٩٢ - الأمــــر (SET)

يستخدم هذا الأمر فى عرض قائمة تتضمن بعض الإختيارات التى تساعد المستخدم على التحكم فى بيئة الحاسب (Environment) والصورة العامة له كالآتى :

```
SET
```

وعند كتابة هذا الأمر أمام مشيرة النقطة يظهر عمود الإختيارات الذى يتضمن الإختيارات الآتية : انظر الشكل (٣٨ - ١٩)



شكل (٣٨ - ١٩)

أ - الإختيار (Options)

ويحتوى هذا الإختيار على إختيارين، الإختيار الأول هو المساعدة (Help) وهو يؤدى إلى عرض شاشات المساعدة التى تظهر مع بعض الأوامر وتقوم بارشاد المستخدم إلى وظائف المفاتيح المختلفة. ويمكن كتابة الأمر الذى يؤدى هذه الوظيفة من مشيرة النقطة (Dot Prompt) ويكون كالآتى (SET HELP ON) كما يمكن إيقاف هذه الشاشات عن طريق كتابة الأمر (SET HELP OFF). والإختيار الثانى هو (Device) ويتم عن طريقه توجيه المخرجات إلى الشاشة (Screen) أو الطابعة (Printer). ويمكن كتابة الأمر الذى يؤدى هذه الوظيفة من مشيرة النقطة (Dot Prompt) ويكون كالآتى (SET DEVICE TO SCREEN) عندما يراد توجيه المخرجات إلى الشاشة أو كالآتى (SET DEVICE TO PRINT) عندما يراد توجيه المخرجات إلى الطابعة.

ب - الإختيار (Screen)

ويستخدم هذا الإختيار للتحكم فى ألوان الشاشة واختيار الألوان المطلوبة للخلفية

(Background) والأعمدة الضوئية (Highlights) وذلك من خلال قائمة الإختيارات الفرعية الخاصة به. ويمكن التحكم فى الألوان أيضا عن طريق مشيرة النقطة (Dot Prompt) وذلك بكتابة الأمر (SET COLOR TO).

جـ - الإختيار (Keys)

ويستخدم هذا الإختيار فى تغيير وظائف المفاتيح المساعدة (Function Keys) التى تظهر فى قائمة الإختيارات الفرعية والموضح بها وظائف هذه المفاتيح مع ملاحظة أن المفتاح (F1) يكون محجوزا للمساعدة (Help) وباقى المفاتيح من (F2) إلى (F10) يتم تغيير وظائفها حسب الحاجة. ويمكن استخدام عددا من الحروف حتى (٣٠) حرفا تمثل وظيفة كل مفتاح. ويمكن تنفيذ هذه العملية من مشيرة النقطة (Dot Prompt) عن طريق الأمر (SET FUNCTION).

مثال

لتخصيص المفتاح (F8) للأمر (CLEAR) من خلال مشيرة النقطة يتم كتابة السطر التالى :

SET FUNCTION 8 To 'CLEAR'

د - الإختيار (Disk)

ويستخدم هذا الإختيار فى معرفة وحدة الأقراص المبدئية (Default) وتحديد وحدة الأقراص التى يتعامل البرنامج مع الملفات الموجودة فيها. ويمكن تنفيذ هذه العملية من خلال مشيرة النقطة عن طريق الأمر (SET DEFAULT TO).

هـ - الإختيار (Files)

ويستخدم هذا الإختيار لفتح مايسمى بالملف البديل (Alternate File) وهو ملف نصوص (Text File) يستخدم فى تخزين الأوامر التى يتم كتابتها من مشيرة النقطة (Dot Prompt). ويمكن فتح هذا الملف من مشيرة النقطة باستخدام الأمر (SET ALTERNATE TO).

و - الإختيار (Margin)

يستخدم هذا الإختيار فى ضبط الهامش الأيسر فى التقارير وكذلك فى حقول الملاحظات (Memo Fields). ويمكن تنفيذ نفس العملية من خلال مشيرة النقطة عن طريق الأمر (SET MARGIN TO).

ز - الإختيار (Decimals)

ويستخدم هذا الإختيار فى تحديد عدد الكسور العشرية المطلوب ظهورها فى الأعداد. ويمكن تنفيذ نفس هذه العملية من خلال مشيرة النقطة عن طريق كتابة الأمر (SET DECIMALS TO).

٩٣ - الأمــــر (SET BELL)

يستخدم هذا الأمر فى التحكم فى الصوت الذى يصدر عند إدخال أى بيانات خطأ أو عند إمتلاء العمود الضوئى الممثل للحقل بالحروف عند إدخال البيانات. والصورة العامة له كالآتى :

SET BELL ON/OFF

والوضع المبدئى (Default) هو (ON).

٩٤ - الأمــــر (SET CATALOG)

يستخدم هذا الأمر فى فتح أو إغلاق ملف الكتالوج. وملف الكتالوج هو ملف يحتوى على جميع ملفات قواعد البيانات والملفات المرتبطة بها والتى يتم إنشاؤها داخل هذا الكتالوج وهو يشبه استخدام الفهارس (Directories) والفهارس الفرعية (Subdirectories) فى نظام التشغيل (MS-DOS). والصورة العامة لهذا الأمر كالآتى :

SET CATALOG ON/OFF

وعند كتابة الأمر (SET CATALOG ON) فإن أى ملفات قواعد بيانات

يتم إنشاؤها تضاف إلى الملفات المخزنة فى الكتالوج المفتوح. أما عند كتابة الأمر (SET CATALOG OFF) فإن أى ملفات جديدة يتم إنشاؤها لاتضاف إلى هذا الكتالوج.

ويجب ملاحظة أن الأمر (SET CATALOG OFF) لايغلق الكتالوج ولكنه يمنع إضافة أى ملفات جديدة إليه ، وذلك عكس الأمر (SET CATALOG TO) دون كتابة أى شيء بعد الأمر ، فإن هذا يؤدى إلى إغلاق ملف الكتالوج وبالتالى عدم القدرة على التعامل مع الملفات المخزنة فيه.

٩٥ - الأمـــر (SET CATALOG TO)

يستخدم هذا الأمر فى فتح كتالوج معين. وهو يختلف عن الأمر (SET CATALOG ON) فى أنه يفتح كتالوج محدد باسمه. أما الأمر (SET CATALOG ON) فإنه يؤدى إلى تجهيز الكتالوج المفتوح لاستقبال أى ملفات أخرى يتم إضافتها. والصورة العامة لهذا الأمر كالآتى :

SET CATALOG TO <filename>

والأمر بهذه الصورة يؤدى إلى فتح الكتالوج الذى يحمل الإسم (filename) إذا كان موجودا وإذا لم يكن موجودا يتم إنشاء كتالوج جديد بهذا الإسم. وإسم الملف يتم كتابته بدون الإمتداد حيث أن البرنامج يضيف الإمتداد (cat.) آليا.

ويستخدم الأمر (SET CATALOG TO) بدون أى شيء بعده فى إغلاق ملف الكتالوج المفتوح وفى هذه الحالة لايمكن التعامل مع الملفات الموجودة فى هذا الكتالوج. وذلك عكس الأمر (SET CATALOG OFF) الذى لايغلق ملف الكتالوج ولكنه يمنع إضافة أى ملفات إليه فقط. ويمكن استخدام علامة الإستفهام مع هذا الأمر كالآتى :

SET CATALOG TO?

وفى هذه الحالة تظهر قائمة بأسماء ملفات الكتالوج الموجودة على القرص الحالى (Current Drive).

وعند استخدام ملف الكتالوج تصبح منطقة العمل العاشرة محجوزة لهذا الكتالوج.

وبذلك يصبح متاحا فقط تسع مناطق عمل (Work Areas) للتعامل مع الملفات من خلالها.

٩٦ - الأمــــر (SET CENTURY)

يستخدم هذا الأمر فى عرض التاريخ أو إدخاله متضمنا الأرقام الدالة على القرن مثل (1955) مثلا بدلا من (55) فقط.

والصورة العامة له كالآتى :

```
SET CENTURY ON/OFF
```

والوضع المبدئى لهذا الأمر هو (OFF) أى عدم ظهور الأرقام الدالة على القرن. ويجب ملاحظة أن عرض هذه الأرقام الإضافية لايؤثر على حجم الحقل الممثل للتاريخ حيث أن حجم حقل التاريخ يكون دائما ثمانية حروف (Bytes).

مثال

```
. ? DATE()
  15/02/90
. SET CENTURY ON
. ? DATE()
  15/02/1990
```

٩٧ - الأمــــر (SET COLOR)

يستخدم هذا الأمر فى مساعدة مخطط البرامج على التحكم فى ألوان الشاشة. وهذا الأمر له صورتان يمكن تلخيصهما كالآتى :

أ - الصورة الأولى

```
SET COLOR ON/OFF
```

وتستخدم هذه الصورة فى التحويل بين الشاشة الملونة (Colored Monitor)

والشاشة الأحادية اللون (Monochrom Monitor). والوضع المبدئى لهذا الأمر هو الوضع الذى يتم تشغيل برنامج (+DBase III) من خلاله. فإذا بدأ التشغيل على شاشة أحادية اللون (Monochrom) يصبح الوضع المبدئى للأمر هو (OFF) وإذا بدأ التشغيل على شاشة ملونة يصبح الوضع المبدئى للأمر هو (ON).

ب - الصورة الثانية

```
SET COLOR TO [<standard>][,<enhanced>]
    [,<border>][,<background>]
```

والإختيار (standard) يقصد به لون الشاشة الخارجية (standard display).

والإختيار (enhanced) المقصود به لون أى أعمدة ضوئية (Highlights) تظهر على الشاشة.

والإختيار (Border) المقصود به الحدود الخارجية للشاشة.

والإختيار (Background) المقصود به لون الخلفية التى تظهر خلف الحروف. والألوان فى كل حالة من هذه الحالات يتم تمثيلها بحروف يمكن تلخيصها فى الجدول التالى :

اللون	الحرف	اللون	الحرف
أسود	N	أحمر	R
أزرق	B	بنفسجى	RB
أخضر	G	بنى	GR
سماوى	BG	أبيض	W
فراغ	X		

وتستخدم علامة (*) مع أى لون لجعله يتلألأ (Blinking) كما تستخدم علامة الجمع(+) مع أى لون للحصول على اللون الأشد إضاءة(High Intensity). فمثلا للحصول على اللون الأصفر - وهو غير موجود فى الجدول - يتم كتابة (+GR) وهو يعنى اللون البنى الشديد الإضاءة أى الأصفر.

مثال

للحصول على شاشة تحتوى على حروف صفراء على خلفية حمراء على أن يكون لون الحروف داخل الأعمدة الضوئية أبيض على خلفية زرقاء مع حدود خضراء يتم كتابة السطر التالى :

SET COLOR TO GR+/R, W/B , G

٩٨ - الأمــــر (SET CONFIRM)

يستخدم هذا الأمر فى شاشات الإدخال ليساعد مخطط البرامج على التحكم فى انتقال المؤشر (Cursor) من حقل إلى آخر حيث أن الوضع المبدئى (Default) أن ينتقل هذا المؤشر إلى الحقل التالى بمجرد إمتلاء الحقل بالحروف. فإذا أريد عدم انتقال المؤشر بعد إمتلاء الحقل يستخدم هذا الأمر. وفى هذه الحالة لايتم الإنتقال إلى الحقل التالى إلا بالضغط على مفتاح الإدخال والصورة العامة لهذا الأمر كالآتى :

SET CONFIRM ON/OFF

٩٩ - الأمــــر (SET CONSOLE)

يستخدم هذا الأمر فى التحكم فى الشاشة من خلال البرنامج عن طريق فتحها أو إغلاقها والصورة العامة له كالآتى :

SET CONSOLE ON/OFF

ويفيد هذا عندما يراد طباعة التقارير دون ظهورها على الشاشة. ويجب مراعاة إعادة الأمر إلى الوضع المبدئى (Default) وهو (ON) قبل نهاية البرنامج.

١٠٠ - الأمــــر (SET DATE)

ويستخدم هذا الأمر فى تغيير صورة التاريخ حسب النظم المختلفة والصورة العامة له كالآتى :

SET DATE AMERICAN/ANSI/BRITISH/ITALIAN/
FRENCH/GERMAN

والوضع المبدئى (Default) هو التاريخ الأمريكى (AMERICAN). والطرق المختلفة لكتابة التاريخ يمكن تلخيصها كالاتى :

```
AMERICAN  = MM/DD/YY
ANSI      = YY.MM.DD
BRITISH   = DD/MM/YY
ITALIAN   = DD-MM-YY
FRENCH    = DD/MM/YY
GERMAN    = DD-MM-YY
```

مثال

لإدخال تاريخ معين فى متغير ذاكرة ثم تحويله إلى الشكل الإنجليزى (BRITISH) يتم كتابة السطور التالية :

```
. mdate = CTOD('01/20/90')
. SET DATE BRITISH
. ? mdate
20/01/90
```

١٠١ - الأمــــر (SET DEBUG ON)

يستخدم هذا الأمر عند اختبار البرنامج لاكتشاف الأخطاء التى قد تكون موجودة به. وهو يرسل خطوات تشغيل البرنامج كما ينفذها الحاسب إلى الطابعة والصورة العامة له كالآتى :

SET DEBUG ON/OFF

وعند كتابة الأمر (SET DEBUG ON) يتم توجيه مخرجات الأمر (SET ECHO ON) إلى الطابعة بدلا من ظهورها على الشاشة.

١٠٢ - الأمــــر (SET DECIMALS)

ويستخدم هذا الأمر لتحديد عدد الأرقام العشرية المطلوب عرضها بالنسبة للمدخلات العددية والصورة العامة له كالآتى :

SET DECIMALS TO <numeric expression>

والوضع المبدئى (Default) هو ظهور رقمين عشريين فى العدد.

١٠٣ - الأمــــر (SET DEFAULT TO)

يستخدم هذا الأمر للإنتقال إلى وحدة أقراص أخرى لتحميل ملفات أو برامج موجودة عليها والصورة العامة له كالآتى :

SET DEFAULT TO <drive>

مثال

عندما يراد تحميل برنامج الطلبة (Cadets) من وحدة الأقراص (B) يتم كتابة السطور التالية :

```
SET DEFAULT TO B
USE Cadets
```

١٠٤ - الأمــــر (SET DELETED)

يستخدم هذا الأمر ليتيح لمخطط البرامج التعامل مع السجلات التى تم وضع علامات عليها لمسحها حيث يمكنه عزل هذه السجلات بحيث لاتؤثر فيها الأوامر التى يتم إدخالها. والصورة العامة له كالآتى :

SET DELETED ON/OFF

وعند كتابة الأمر (SET DELETED ON) فإن البرنامج يتعامل مع هذه السجلات كأنها غير موجودة.

١٠٥ - الأمـــر (SET DEVICE)

يستخدم هذا الأمر فى توجيه السطور التى يتم كتابتها بواسطة الأمر (@...SAY) إلى الشاشة أو الطابعة حسب الحاجة. والصورة العامة له كالآتى :

SET DEVICE TO PRINT/SCREEN

والوضع المبدئى (DEFAULT) هو توجيه المخرجات إلى الشاشة. وعند كتابة الأمر (SET DEVICE TO PRINT) فإن مخرجات الأمر (@...SAY) يتم توجيهها إلى الطابعة أما مخرجات الأمر (@...GET) فإنها لاتذهب إلى الطابعة.

١٠٦ - الأمـــر (SET DOHISTORY)

يستخدم هذا الأمر للتحكم فى تخزين الأوامر فى مخزن التاريخ (History) أو عدم تخزينها فيه حسب الحاجة والصورة العامة له كالآتى :

SET DOHISTORY ON/OFF

ويجب ملاحظة أن هذا الأمر يؤثر فقط فى الأوامر التى يتم كتابتها فى البرنامج ولكنه لايؤثر فى الأوامر المباشرة التى يتم إدخالها عن طريق مشيرة النقطة (Dot Prompt) حيث أن هذه الأوامر يتم تخزينها فى مخزن التاريخ (History) دون الحاجة إلى كتابة هذا الأمر. ويستخدم هذا الأمر عادة عند اختبار البرنامج وتصحيحه.

١٠٧ - الأمـــر (SET ECHO)

يستخدم هذا الأمر فى عرض أوامر البرنامج أثناء تنفيذها والصورة العامة له كالآتى :

SET ECHO ON/OFF

والوضع المبدئى (Default) هو (Off) أى عدم ظهور الأوامر أثناء تنفيذها. وهو يستخدم عادة عند اختبار البرنامج وتصحيحه.

١٠٨ - الأمـــر (SET ESCAPE)

يستخدم هذا الأمر للتحكم فى إيقاف البرنامج أو استمراره نتيجة للضغط على مفتاح الهروب (ESC) والصورة العامة له كالآتى :

SET ESCAPE ON/OFF

والوضع المبدئى هو (ON) وهذا يعنى أن الضغط على مفتاح الهروب يؤدى إلى إيقاف تنفيذ البرنامج.

١٠٩ - الأمـــر (SET EXACT ON)

يستخدم هذا الأمر فى التحكم فى مقارنة البيانات الحرفية حيث أن المقارنة فى الوضع المبدئى (Default) تتم بين حروف السلسلة الحرفية الأولى والحروف المقابلة لها من السلسلة الحرفية الثانية فإذا إنتهت السلسلة الأولى أصبحت السلسلتان متطابقتين بالرغم من أنهما فعليا قد يكونان غير متطابقتين تماما. ولذلك يستخدم هذا الأمر فى تحويل المقارنة إلى مقارنة تامة (Exact) أى تطابق السلسلتين تماما. والصورة العامة لهذا الأمر كالآتى :

SET EXACT ON/OFF

مثـــال

عند كتابة السطر التالى :

? 'abc' = 'abcdef'

يظهر على الشاشة النتيجة (.F.) أى غير صحيح (False).

وعند كتابة السطر التالى :

```
? 'abcdef' = 'abc'
```

يظهر على الشاشة النتيجة (.T.) أى صحيح (True) وذلك لأن البرنامج يقارن بين السلسلتين حتى إنتهاء السلسلة اليمنى.

وعند كتابة السطرين التاليين :

```
SET EXACT ON
? 'abcdef' = 'abc'
```

يظهر على الشاشة القيمة (.F.) أى غير صحيح (False) وذلك لأن البرنامج يطابق السلسلتين مطابقة كاملة.

١١٠ - الأمـــر (SET FIELDS)

يستخدم هذا الأمر عندما يراد التحكم فى الحقول التى سبق تحديدها بواسطة الأمر (SET FIELDS TO) حيث يتيح لمخطط البرامج استخدام هذه الحقول أو عدم استخدامها حسب الحاجة والصورة العامة له كالآتى :

```
SET FIELDS ON/OFF
```

والوضع المبدئى (Default) هو (OFF) وهو يعنى أن جميع حقول قاعدة البيانات يمكن التعامل معها حتى لو سبق تحديد بعض الحقول بالأمر (SET FIELDS TO). أما باستخدام الأمر (SET FIELDS ON) فإن الحقول التى سبق تحديدها فقط هى التى يمكن استخدامها وفى هذه الحالة يمكن التعامل مع الحقول الموجودة فى منطقة العمل الحالية (Current work area) أو التعامل مع الحقول الموجودة فى مناطق العمل الأخرى فى نفس الوقت.

١١١ - الأمـــر (SET FIELDS TO)

يستخدم هذا الأمر فى تحديد الحقول المطلوب التعامل معها فى ملف أو أكثر من ملفات قاعدة البيانات. والصورة العامة له كالآتى :

SET FIELDS TO [<field list>/ALL]

وعندما يراد إضافة حقول من مناطق عمل أخرى غير منطقة العمل الحالية (Current work area) يستخدم الإسم المرادف (Aias) الخاص بهذه المنطقة يليه العلامة (->) التى يتم تكوينها من علامة الناقص (-) وعلامة أكبر من (>) ويلى ذلك إسم الحقل المطلوب إدخاله ضمن قائمة الحقول المطلوبة.

تحذير

عند استخدام الأمر (SET FIELDS ON) دون تحديد الحقول المطلوبة أولا باستخدام الأمر (SET FIELDS TO) لايمكن التعامل مع الحقول. ويلزم فى هذه الحالة استخدام الأمر (SET FIELDS OFF) للعودة إلى الوضع المبدئى (Default).

ملاحظة

كما سبق الإيضاح يمكن استخدام الأمر (SET FIELDS TO) فى التعامل مع حقول من ملفات فى عدة مناطق عمل فى نفس الوقت ولكن هذه الحقول لاتكون مرتبطة ببعضها أى لايمكن عرض بيانات سجل واحد مثلا يتكون من عدة حقول من مناطق عمل مختلفة. والطريقة الوحيدة لربط هذه الحقول هى استخدام الأمر(SET RELATION TO) فى ربط الملفات مع بعضها.

١١٢ - الأمـــر (SET FILTER)

يستخدم هذا الأمر فى ترشيح أو تصفية قاعدة البيانات بحيث تبدو كأنها لاتحتوى إلا على السجلات التى تحقق الشروط الموجودة فى المرشح (Filter) والصورة العامة لهذا الأمر كالآتى :

SET FILTER TO [FILE <filename> /?][<condition>]

ويمكن استخدام الأمر (SET FILTER TO) دون كتابة أى شيء بعده وفى هذه الحالة يتم إغلاق المرشح الذى سبق فتحه لملف قاعدة البيانات المفتوح.

والإختيار (FILE <filename>) يستخدم عندما يراد استخدام ملف البحث (Query File) الذى سبق إنشاؤه بواسطة الأمر (CREATE/MODIFY QUERY). وتستخدم علامة الإستفهام (?) بدلا من إسم الملف لعرض أسماء جميع ملفات البحث الخاصة بملف قاعدة البيانات المفتوح وذلك لاختيار الملف المطلوب منها.

والاختيار (Condition) يستخدم فى إدخال الشرط أو الشروط المطلوب استخدامها فى ترشيح قاعدة البيانات.

مثـــال

إذا أريد التعامل مع السجلات الخاصة بالطلبة الذين التحقوا بالمعهد بعد (١٩٨٥/١/١) مثلا يتم كتابة السطر التالى :

SET FILTER TO dat_ent > CTOD "01/01/85"

حيث (dat_ent) هو إسم الحقل الذى يمثل تاريخ إلتحاق الطالب فى المعهد وفى هذه الحالة يتعامل البرنامج مع قاعدة البيانات كأنها لاتحتوى إلا على بيانات الطلبة الذين يحققون هذا الشرط.

١١٣ - الأمــــر (SET FIXED)

يستخدم هذا الأمر عندما يراد عرض البيانات العددية محتوية على عدد ثابت من الأرقام العشرية. وهذا العدد يكون فى الوضع المبدئى رقمين وذلك فى حالة عدم استخدام الأمر (SET DECIMALS). والصورة العامة لهذا الأمر هى :

SET FIXED ON / OFF

١١٤ - الأمــــر (SET FORMAT)

يستخدم هذا الأمر فى فتح ملف التشكيل (Format File) المطلوب استخدامه فى إدخال البيانات إلى ملف قاعدة البيانات. والصورة العامة لهذا الأمر كالآتى :

SET FORMAT TO [<filename>/?]

ويتم كتابة إسم الملف (filename) بدون الإمتداد حيث أن البرنامج يفترض أن الإمتداد (fmt.) فى هذه الحالة. ويمكن استخدام علامة الإستفهام (?) بدلا من إسم الملف لعرض أسماء ملفات التشكيل الخاصة بملف قاعدة البيانات المفتوح واختيار الملف المطلوب منها. ويمكن إغلاق ملف التشكيل باستخدام الأمر (CLOSE FORMAT) أو الأمر (SET FORMAT TO) دون كتابة أى شيء بعده.

١١٥ - الأمــــر (SET FUNCTION)

يستخدم هذا الأمر فى تخصيص وظائف لمفاتيح الوظائف (Function Keys). وفى هذه الحالة يتم تغيير وظيفة المفتاح السابقة بالوظيفة الجديدة التى يتم كتابتها والصورة العامة لهذا الأمر كالآتى :

SET FUNCTION <expl> TO <exp2> [;]

حيث (exp1) هو الرقم الممثل للمفتاح المطلوب تغيير وظيفته و (exp2) هو إسم الوظيفة المطلوب إدخالها وحرف الفاصلة المنقوطة (;) يمثل مفتاح الإدخال وذلك لكى يتم تنفيذ الوظيفة المطلوبة بمجرد الضغط على المفتاح الخاص بها دون الحاجة إلى الضغط على مفتاح الإدخال.

ويمكن إدخال إسم الوظيفة المطلوبة (exp2) حتى (٣٠) حرفا. ويجب ملاحظة أنه يمكن تغيير وظائف المفاتيح العشرة ماعدا مفتاح (F1) لأنه يكون محجوزا لتشغيل شاشات المساعدة (Help).

مثـال

لتخصيص المفتاح (F5) للأمر (DISPLAY) يتم كتابة السطر التالى :

SET FUNCTION 5 TO 'DISPLAY;'

ولتخصيص المفتاح (F8) لتنفيذ عدة وظائف يمكن كتابة السطر التالى :

SET FUNCTION 8 TO 'CLEAR;USE CADETS INDEX NAME;'

١١٦ - الأمــــر (SET HEADING)

يستخدم هذا الأمر عندما يراد التحكم فى عناوين الحقول التى تظهر مع الأوامر (DISPLAY, LIST, SUM, AVERAGE) وذلك بعرضها أو عدم عرضها حسب الحاجة. والصورة العامة لهذا الأمر كالآتى :

SET HEADING ON/OFF

والوضع المبدئى لهذا الأمر هو (ON) وفى بعض الأحيان يريد مخطط البرامج إدخال العناوين التى يريدها باستخدام الأمر (@...SAY) ، فى هذه الحالة يقوم باستخدام الأمر (SET HEADING OFF).

مثال

عندما يراد عرض سجل مثلا من قاعدة بيانات الطلبة يشمل الحقلين (name) ، (address) يتم كتابة السطور التالية :

USE Cadets
DISPLAY name, address

وعند الضغط على مفتاح الإدخال يظهر الآتى :

Record #	name	address
1	Hatem Mahmoud	20 Tahreer street

وإذا أريد عدم عرض عناوين الحقول تستخدم الأوامر التالية :

```
USE Cadets
SET HEADING OFF
DISPLAY name, address
```

وعند الضغط على مفتاح الإدخال يظهر الآتى :

```
1    Hatem Mahmoud    20 Tahreer street
```

١١٧ - الأمـــر (SET HISTORY)

يستخدم هذا الأمر فى تشغيل مخزن التاريخ (History) أو عدم تشغيله حسب الحاجة والصورة العامة له كالآتى :

```
SET HISTORY ON/OFF
```

والوضع المبدئى لهذا الأمر هو (ON) وهذا يسمح باسترجاع الأوامر التى سبق إدخالها من مشيرة النقطة (Dot Prompt) باستخدام مفتاح السهم لأعلى ومفتاح السهم لأسفل. وبالتالى يمكن تنفيذ الأمر المطلوب بالضغط على مفتاح الإدخال فقط دون الحاجة إلى كتابة هذا الأمر مرة أخرى.

١١٨ - الأمـــر (SET HISTORY TO)

يستخدم هذا الأمر فى تحديد عدد الأوامر التى يمكن تخزينها فى مخزن التاريخ والصورة العامة له كالآتى :

```
SET HISTORY TO <exp>
```

حيث (exp) هو عدد الأوامر المطلوب تحديده والعدد المبدئى (Default) هو عشرون والعدد المسموح بإدخاله هو أى عدد من صفر إلى ١٦ ألف أمر حسب الذاكرة المتاحة.

١١٩ - الأمـــــر (SET INDEX)

يستخدم هذا الأمر فى فتح ملفات الفهرس لاستخدامها فى ترتيب السجلات والصورة العامة له كالآتى :

SET INDEX TO [<list of index filenames>/?]

ويتم كتابة إسم الملف بدون الإمتداد حيث أن البرنامج يضيف الإمتداد (ndx.) آليا. ويمكن إغلاق ملفات الفهرس المفتوحة عن طريق كتابة الأمر (SET INDEX TO) دون كتابة أى شىء بعده كما يمكن إغلاقها أيضا باستخدام الأمر (CLOSE INDEX). وتستخدم علامة الإستفهام (?) بدلا من إسم الملف لعرض أسماء جميع ملفات الفهرس الخاصة بملف قاعدة البيانات المفتوح. ويمكن فتح حتى سبعة ملفات فهرس فى نفس الوقت. ويجب ملاحظة أن هذا الأمر يفتح ملف الفهرس فقط ولكنه لاينشؤه ويصبح أول ملف فهرسى يتم فتحه هو الملف الرئيسى (Master). وعند إجراء أى تعديل فى بيانات ملف قاعدة البيانات يتم تحديث ملفات الفهرس المفتوحة تبعا لهذا التعديل.

مثـال

عندما يراد فتح ملفات الفهرس الخاصة بملف العملاء (Clients) يتم كتابة السطور التالية :

```
USE Clients
SET INDEX TO Job, Name
```

وهذا يعنى أن ملف الفهرس (Job) هو الملف الرئيسى (Master).

١٢٠ - الأمـــــر (SET MARGIN)

يستخدم هذا الأمر لضبط الهامش الشمال للتقارير المطبوعة والصورة العامة له كالآتى :

SET MARGIN TO <exp>

حيث (exp) هو قيمة عددية تمثل هذا الهامش.

مثـال

SET MARGIN TO 10

١٢١ - الأمـــــر (SET MEMOWIDTH)

يستخدم هذا الأمر فى تحديد عرض بيانات حقل الملاحظات (memo field) عند عرضها على الشاشة أو طباعتها والصورة العامة له كالآتى :

SET MEMOWIDTH TO <exp>

والوضع المبدئى لهذا العرض هو (٥٠) حرفا.

١٢٢ - الأمـــــر (SET MENU)

يستخدم هذا الأمر للتحكم فى ظهور أو عدم ظهور شاشات المساعدة التى تظهر للمستخدم لتعريفه بوظائف مفاتيح التحكم فى المؤشر لتصحيح المدخلات قبل إدخالها. والصورة العامة لهذا الأمر كالآتى :

SET MENU ON/OFF

والوضع المبدئى هو (ON).

١٢٣ - الأمـــــر (SET ORDER)

يستخدم هذا الأمر فى التحكم فى ترتيب ملفات الفهرس المفتوحة وبالتالى تحديد أيها تصبح ملفا رئيسيا (Master). والصورة العامة له كالآتى :

SET ORDER TO [<exp>]

حيث : (exp) هو رقم صحيح يأخذ القيمة من صفر إلى (٧) حسب عدد ملفات

الفهرس المفتوحة. وعند كتابة القيمة صفر يعود الملف إلى وضعه الأول دون فهرسة وذلك دون إغلاق ملفات الفهرس. وعند كتابة أى رقم آخر فإن هذا الرقم يمثل ترتيب ملف الفهرس الذى يصبح هو الملف الرئيسى (Master).

مثال

لفتح ملفات الفهرس الخاصة بقاعدة بيانات الموظفين (Clients) يتم كتابة السطرين التاليين :

```
USE Clients
SET INDEX TO Job, Name
```

وعندما يراد استخدام ملف (Name.ndx) كفهرس رئيسى يتم كتابة السطر التالى :

```
SET ORDER TO 2
```

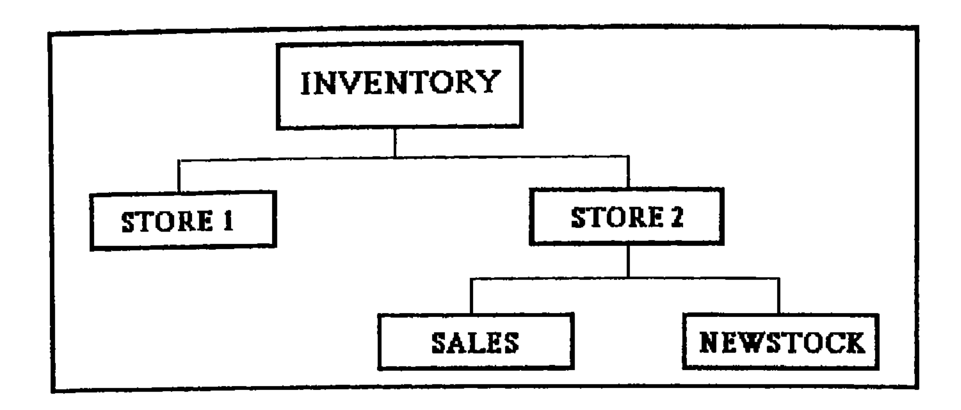
١٢٤ - الأمـــر (SET PATH)

يستخدم هذا الأمر فى تحديد مسار معين للملفات المطلوب استخدامها والصورة العامة له كالآتى :

```
SET PATH TO  [<path list>]
```

واستخدام الأمر (SET PATH TO) دون كتابة أى شيء بعده يعيد المسار إلى الفهرس الفرعى الحالى (Current Directory). وتحديد المسار يساعد البرنامج على البحث عن الملفات المطلوبة وذلك عن طريق تنظيمها على شكل شجرة (Tree) حيث يبدأ البحث من الجذر (Root) وينتقل إلى الفروع والفروع الأصغر وهكذا. والمسار عبارة عن فهارس فرعية يفصلها الشرطة المائلة (\ ) ويبدأ بالفهرس الرئيسى (Root Directory) يليه الفهارس الفرعية الأخرى. وقائمة المسارات (Path List) هى مجموعة من المسارات (Pathes) يفصل بينها فاصلة أو فاصلة منقوطة (;). وعندما يبحث البرنامج عن أى ملف فإنه يبحث فى المسار الأول فإذا لم يجده ينتقل إلى المسار الثانى ثم الثالث ..... وهكذا.

انظر شكل (٣٨ - ٢٠)



شكل (٣٨ - ٢٠)

مثال

إذا كان الملف (Cadets.prg) غير موجود فى الفهرس الحالى فإن الأوامر التالية تؤدى إلى البحث عنه فى مسارات أخرى كالآتى :

```
SET PATH TO A:\DBase, B:\Hasan
DO Cadets
```

وفى هذه الحالة يتم البحث فى الفهرس الفرعى (\DBase) أولا ثم يتم الإنتقال إلى الفهرس الفرعى (\Hasan) حتى يتم الوصول إلى الملف المطلوب.

١٢٥ - الأمـــر (SET PRINT)

يستخدم هذا الأمر فى توجيه المخرجات التى لايتم كتابتها باستخدام الأمر (@...SAY) إلى الطابعة مع ظهورها على الشاشة فى نفس الوقت وهى البيانات الناتجة من استخدام الأوامر (? , DISPLAY , LIST). والصورة العامة له كالآتى :

```
SET PRINT ON/OFF
```

والوضع المبدئى له (OFF).

١٢٦ - الأمـــــر (SET PROCEDURE)

يستخدم هذا الأمر فى فتح ملف الخطوات أو الإجراءات (Procedure File). والصورة العامة له كالآتى :

SET PROCEDURE TO [<filename>]

وإسم الملف يجب أن يتضمن رمز وحدة الأقراص إذا كانت غير وحدة الأقراص الحالية (Current Drive). ولايشترط إضافة الإمتداد حيث أن البرنامج يضيف الإمتداد (prg.) آليا. ويتم إغلاق ملف الخطوات أو الإجراءات عن طريق كتابة الأمر (SET PROCEDURE TO) دون كتابة أى شىء بعده.

وملف الخطوات يمكنْ أن يحتوى على (٣٢) برنامج خطوات (Procedure). ويتم تحديد بداية كل برنامج عن طريق كتابة الأمر (PROCEDURE) فى أوله. ويجب ملاحظة أنه لايمكن فتح أكثر من ملف خطوات فى نفس الوقت. وإذا أريد استخدام عدة ملفات يتم إغلاق الملف السابق وفتح ملف جديد وهكذا.

١٢٧ - الأمـــــر (SET RELATION)

يستخدم هذا الأمر فى ربط ملفين من ملفات قواعد البيانات باستخدام حقل مشترك بين الملفين والصورة العامة له كالآتى :

SET RELATION TO[<keyfield>/<RECNO()>
INTO <alias>]

ويقوم هذا الأمر بربط ملف قاعدة البيانات الموجود فى منطقة العمل الحالية (Current Work Area) بملف آخر مفتوح فى منطقة عمل أخرى. وهذا الملف الآخر يتم تحديده عن طريق الإختيار (alias) أو المرادف الذى يشمل إسم منطقة العمل وإسم الملف المفتوح فيها. ويمكن ربط الملفين عن طريق حقل مشترك فيهما (keyfield) وفى هذه الحالة يجب أن يكون الملف المربوط مفهرسا (Indexed) بناء على هذا الحقل المشترك.

كما يمكن ربط الملفين عن طريق رقم السجل (RECNO()) وفى هذه الحالة يجب أن يكون الملف المربوط غير مفهرس (Unindexed). ويمكن تخزين هذه العلاقة فى ملف منظر(VIEW FILE) بواسطة الأمر(CREATE VIEW FROM ENVIRONMENT).

مثال

إذا كان هناك ملفان لبيانات الطلبة (Cadets1) ، (Cadets2) ويراد ربطهما بناء على الحقل المشترك بينهما وهو حقل الإسم (name) يتم كتابة الأوامر التالية :

```
SELECT 1
USE Cadets1
SELECT 2
USE Cadets2 INDEX name
SELECT 1
SET RELATION TO name INTO cadets2
```

ويلاحظ أن الملف المطلوب ربطه وهو (Cadets2) تم فهرسته بناء على الحقل المشترك بين الملفين وهو حقل الإسم (name). كما يلاحظ أن آخر منطقة عمل تم اختيارها بواسطة الأمر (SELECT) هى منطقة العمل التى يتم إنشاء العلاقة داخلها.

١٢٨ - الأمــــر (SET SAFETY)

يستخدم هذا الأمر فى تأمين الملفات عن طريق منع النسخ فوقها ويحدث هذا عندما يريد المستخدم نسخ ملف بإسم ملف موجود. فى هذه الحالة تظهر الرسالة التالية :

```
<filename> already exists, overwrite it ? Y/N
```

وفى بعض الأحيان يريد مخطط البرامج التحكم فى نسخ الملفات دون ظهور هذه الرسالة التى تسبب توقف البرنامج ولذلك يقوم باستخدام هذا الأمر والصورة العامة له كالآتى :

```
SET SAFETY ON/OFF
```

والوضع المبدئى لهذا الأمر هو (ON).

١٢٩ - الأمـــر (SET SCOREBOARD)

يستخدم هذا الأمر عندما يراد استخدام السطر رقم صفر الذى يكون محجوزا عادة لرسائل برنامج (+DBase III) فى حالة استخدام الأمر (SET STATUS OFF) والصورة العامة له كالآتى :

SET SCOREBOARD ON/OFF

وهو يكون عادة (ON).

١٣٠ - الأمـــر (SET STATUS OFF)

يستخدم هذا الأمر للتحكم فى ظهور عمود الحالة (Status Bar) أو عدم ظهوره والصورة العامة له كالآتى :

SET STATUS ON/OFF

والوضع المبدئى له (ON) ويتيح هذا الأمر لمخطط البرامج استخدام السطر رقم (٢٢) الذى يكون محجوزا لعمود الحالة فى عرض بيانات على الشاشة أو تصميم شاشات الإدخال.

١٣١ - الأمـــر (SET STEP)

يستخدم هذا الأمر فى تنفيذ خطوات البرنامج خطوة خطوة مع الوقوف بعد كل خطوة ويساعد هذا على اكتشاف أخطاء البرنامج إن وجدت. والصورة العامة لهذا الأمر كالآتى :

SET STEP ON/OFF

والوضع المبدئى له (OFF).

١٣٢ - الأمـــر (SET TALK)

يستخدم هذا الأمر فى التحكم فى عرض خطوات تنفيذ البرنامج على الشاشة أوعدم عرضها. والصورة العامة له كالآتى :

SET TALK ON/OFF

والوضع المبدئى له (ON) وعادة يحوله مخطط البرامج إلى (OFF) عن طريق كتابة الأمر (SET TALK OFF) فى بداية البرنامج حتى يتحكم فيما يعرض على الشاشة أو على الطابعة.

مثـال

يمكن ملاحظة الفرق بين الوضع (ON) والوضع (OFF) مع هذا الأمر من خلال السطور التالية :

```
.  STORE 'Mohamed' TO mname1
Mohamed
.  SET TALK OFF
.  STORE 'Hasan' TO mname2
```

يلاحظ بعد السطر الأول ظهور محتويات متغير الذاكرة (mname) نتيجة تنفيذ الأمر أما بعد السطر الأخير فلاتظهر محتويات متغير الذاكرة نتيجة استخدام الأمر (SET TALK OFF).

١٣٣ - الأمـــر (SET TYPEAHEAD)

يستخدم هذا الأمر فى تحديد عدد الحروف المسموح بالإحتفاظ بها فى مخزن الذاكرة (Buffer) قبل إنتقالها إلى الذاكرة الداخلية والصورة العامة له كالآتى :

SET TYPEAHEAD TO <exp>

حيث (exp) هو عدد الحروف المطلوب تحديده والعدد المبدئى يكون عشرين حرفا. وهذا الأمر يعمل فقط عندما يكون الأمر (SET ESCAPE) على الوضع

(ON) أى أنه لايعمل عند استخدام الأمر (SET ESCAPE OFF).
وكذلك لايعمل عند استخدام أى أمر من الأوامر (ON KEY) ،
(INKEY) حيث أن هذا يؤدى إلى إدخال أى حرف يكتبه المستخدم إلى
الذاكرة الداخلية مباشرة.

١٣٤ - الأمـــر (SET UNIQUE)

يستخدم هذا الأمر للتحكم فى إدخال السجلات التى تشترك فى نفس قيمة الحقل الفهرسى أو عدم إدخالها فى الفهرس والصورة العامة له كالآتى :

SET UNIQUE ON/OFF

والوضع المبدئى له (OFF).

وعند إنشاء ملف فهرس مع استخدام الأمر (SET UNIQUE ON) فإن السجلات التى تحتوى على نفس القيمة للحقل الفهرسى لايتم إدخالها فى الفهرس ولكن يتم إدخال أول سجل فقط وفى هذه الحالة لايكون الملف الفهرسى محتويا على أرقام سجلات مشتركة فى قيمة الحقل الفهرسى. واستخدام هذا الأمر قبل فتح ملف الفهرس (Index File) يؤدى نفس العمل الذى يؤديه الأمر :

INDEX ON <key expression> TO <filename> UNIQUE

وعند إضافة أو تعديل أى سجلات فى الملف فإن الملف يستعيد الحالة المنفردة (Unique) أى أن إضافة أى سجلات تحتوى على حقل فهرسى يماثل الحقل الفهرسى لسجلات أخرى موجودة فى الملف لايؤدى إلى إضافة أرقام هذه السجلات فى الفهرس باعتبارها سجلات متكررة.

١٣٥ - الأمـــر (SET VIEW TO)

يستخدم هذا الأمر فى فتح ملف المنظر (View File). والصورة العامة له كالآتى :

SET VIEW TO <filename>/?

ويتم إدخال إسم الملف (filename) بدون الإمتداد حيث أن البرنامج يفترض الإمتداد (.vue) ، وتستخدم علامة الإستفهام (?) لعرض أسماء جميع ملفات المنظر المخزنة على القرص أو فى الكتالوج المفتوح.

١٣٦ - الأمـــر (SKIP)

يستخدم هذا الأمر فى نقل مؤشر السجلات (Record Pointer) إلى سجل تال أو سجل سابق خلال ملف قاعدة البيانات المفتوح والصورة العامة له كالآتى :

SKIP [<exp>]

حيث (exp) هو عدد السجلات المطلوب تحريك المؤشر خلالها ويتحرك المؤشر إلى السجلات التالية بما يساوى هذا العدد. كما يمكن أن يتحرك إلى السجلات السابقة عند كتابة علامة ناقص (-) قبل العدد (exp).
وإذا لم يتم إضافة أى عدد إلى الأمر فإنه يؤدى إلى تحريك المؤشر سجلا واحدا إلى الأمام.

أمثـلة

لتحريك المؤشر إلى السجل الثانى فى ملف قاعدة بيانات الطلبة (Cadets) يتم كتابة الأوامر التالية :

```
USE Cadets
SKIP
```

ولتحريك المؤشر من السجل الثانى إلى السجل رقم (٩) يتم كتابة السطر التالى :

```
SKIP 7
```

وللرجوع إلى السجل رقم (٦) يتم كتابة السطر التالى :

```
SKIP -3
```

ويمكن تخزين عدد معين فى متغير ذاكرة واستخدام متغير الذاكرة مع الأمر (SKIP) وذلك كالآتى :

```
STORE 2 TO mskip
SKIP mskip
```

١٣٧ - الأمـــر (SORT)

يستخدم هذا الأمر فى إنشاء ملف قاعدة بيانات يحتوى على سجلات الملف الأصلى مرتبة بالترتيب المطلوب تبعا لمحتويات حقل معين أو عدة حقول والصورة العامة له كالآتى :

```
SORT [<scope>] TO [<filename>] ON <field1>
   [/A][/D][/C] [,<field2>][/A][/D][/C] ....
   [WHILE <condition>][FOR <condition]
```

والإختيار (scope) يستخدم لتحديد المدى فى الملف المطلوب ترتيبه حيث يمكن ترتيب الملف كله (ALL) أو مجموعة من السجلات إبتداء من سجل معين (REST). والإختيار (filename) هو إسم ملف قاعدة البيانات المرتب الذى سيتم إنشاؤه.

والإختيارات (field1) ، (field2) تستخدم لتحديد الحقل أو الحقول التى يتم الترتيب بناء عليها. ويمكن استخدام حتى عشرة حقول فى ترتيب الملف ، والاختيار (/A) يستخدم للترتيب تصاعديا (Ascendingly) ويجب ملاحظة أن الوضع المبدئى هو الترتيب تصاعديا.

والإختيار (/D) يستخدم للترتيب تنازليا (Descendingly) والإختيار (/C) يستخدم عندما يراد عدم التمييز بين الحروف الكبيرة (Capital) أو الصغيرة (Small) فى الترتيب. كما يستخدم الاختياران (WHILE) و (FOR) فى تحديد شروط البحث التى يتم من خلالها اختيار السجلات المطلوب ترتيبها.

مثال

يمكن ترتيب ملف الطلبة (Cadets) بناء على حقلى المهنة (Job) والإسم (name) كالآتى :

SORT ON Job, name TO scadets

حيث يصبح الملف (scadets) ملف قاعدة بيانات آخر يحتوى على نفس سجلات الملف الأصلى (Cadets) مرتبة حسب حقلى المهنة والإسم على الترتيب.

١٣٨ - الأمـــر (STORE)

يستخدم هذا الأمر فى إنشاء متغير أو عدة متغيرات ذاكرة (memory variables) والصورة العامة له كالآتى :

STORE <exp> TO <memory variable list>

حيث (exp) هى القيمة أو القيم التى يتم تخزينها فى متغيرات الذاكرة. ويمكن تخزين قيمة واحدة فى عدة متغيرات يتم تحديدها من خلال الإختيار (memory variable list).

وهناك طريقة أخرى يمكن عن طريقها إنشاء متغير الذاكرة وذلك كالآتى :

<memory variable > = <expression>

ولكن هذه الطريقة لاتسمح بإنشاء عدة متغيرات فى سطر واحد مثل الأمر (STORE).

ملاحظة

الحقل الذى يحمل نفس إسم متغير الذاكرة تكون له الأسبقية عند تنفيذ أى عملية على هذا الحقل. فمثلا إذا كان هناك حقل يحمل الإسم (name) ومتغير ذاكرة يحمل نفس الإسم (name) فإن أى عملية يتم إجراؤها على متغير الذاكرة لاتؤثر فيه ولكنها تؤثر فى الحقل (name) فقط.

مثال

لتخزين القيمة صفر فى عدة متغيرات (a, b, c) يتم كتابة السطر التالى :

STORE 0 TO a, b, c

١٣٩ - الأمــــر (SUM)

يستخدم هذا الأمر فى تجميع الحقول العددية فى ملف قاعدة البيانات المفتوح والصورة العامة له كالآتى :

SUM [<scope>][<fields>][TO <memvar list>]
[WHILE <condition>][FOR <condition>]

ويلاحظ أن جميع الإختيارات اختيارية أى يمكن كتابة الأمر دون كتابة أى شىء بعده وفى هذه الحالة يتم تجميع جميع الحقول العددية لجميع سجلات قاعدة البيانات.

والإختيار (scope) يتم من خلاله تحديد المدى الذى يتم البحث فيه عن السجلات المطلوب تجميع بيانات الحقول العددية فيها.

والإختيار (fields) يستخدم لتحديد الحقول المطلوب تجميعها.

والإختيار (<TO <memvar list) يستخدم عندما يراد تخزين هذا المجموع فى متغيرات ذاكرة يتوقف عددها على عدد الحقول المطلوب تجميعها.

كما يستخدم الإختياران (WHILE) و (FOR) للبحث عن السجلات المطلوب تجميع بياناتها العددية.

مثـال

عندما يراد تجميع حقلى المرتب (Salary) والساعات (Hours) فى ملف الموظفين (Clients) يتم كتابة السطر التالى :

SUM salary , hours TO msalary , mhours

١٤٠ - الأمــــر (SUSPEND)

يستخدم هذا الأمر فى إيقاف تنفيذ أوامر البرنامج إيقافا مؤقتا. ويمكن كتابة

هذا الأمر داخل البرنامج ليساعد مخطط البرامج على إيقاف البرنامج فى المكان الذى يشك فى وجود خطأ فيه حيث يستطيع عرض الأوامر السابقة لهذا المكان والمخزنة فى مخزن التاريخ (History) ثم يمكنه العودة مرة ثانية إلى تنفيذ أوامر البرنامج عن طريق الأمر (RESUME).

١٤١ - الأمـــر (TEXT)

يستخدم هذا الأمر فى كتابة نصوص كبيرة على الشاشة أو الطابعة والصورة العامة له كالآتى :

```
TEXT
<text characters>
ENDTEXT
```

ويتم كتابة النص بين (TEXT) و (ENDTEXT). ويمكن استخدام هذا الأمر فى عرض قوائم الإختيارات الكبيرة أو شاشات المساعدة (Help).

مثـال

يمكن كتابة السطور التالية فى البرنامج :

```
TEXT
this is an example of text that is to be displayed on the
screen as it is.
ENDTEXT
```

١٤٢ - الأمـــر (TOTAL)

يستخدم هذا الأمر فى تجميع بيانات الحقول العددية لكل السجلات أو لمجموعة من السجلات التى تحقق شروطا معينة ويتم التجميع لكل مجموعة من السجلات تشترك فى قيمة الحقل الذى يستخدم كمفتاح (Key Field) لتقسيم هذه المجموعات. والصورة العامة لهذا الأمر كالآتى :

```
TOTAL ON <keyfield> TO <filename>[<scope>]
  [FIELDS <fieldlist>][WHILE <condition>]
  [FOR <condition>]
```

والإختيار (keyfield) يستخدم فى تحديد المجموعات التى يتم تجميع الحقول العددية بها وهو أحد حقول ملف قاعدة البيانات.

وإسم الملف (filename) هو ملف قاعدة بيانات جديد يتم إنشاؤه متضمنا مجموع البيانات العددية للمجموعات المختلفة من السجلات.

والإختيار (scope) يستخدم لتحديد مدى السجلات المطلوب البحث خلاله عن السجلات المطلوب إدخالها فى الملف الجديد.

والإختيار (FIELDS <field list>) يستخدم لتحديد أسماء الحقول المطلوب إدخالها فى الملف الجديد.

ويستخدم الإختياران (WHILE) و (FOR) فى تحديد السجلات المطلوب إدخالها فى الملف الجديد حسب شروط البحث التى يتم إدخالها.

ويجب ملاحظة أن ملف قاعدة البيانات المفتوح والذى يتم تجميع بياناته يجب أن يكون مفهرسا (Indexed) أو مفروزا (Sorted) بناء على الحقل المستخدم كمفتاح (Key Field).

ملاحظة

تركيب الملف الجديد (Structure) يكون مماثلا تماما لتركيب ملف قاعدة البيانات الأصلى باختلاف واحد وهو عدم نسخ حقول الملاحظات (memo fields) فى الملف الجديد.

مثال

لتجميع بيانات حقول ملف قاعدة بيانات الموظفين (Employees) تبعا لحقل المهنة (Job) يتم كتابة السطور التالية :

```
USE Employees
INDEX ON Job TO Job
TOTAL ON Job TO Job
```

ويلاحظ من السطر الثانى أنه تم استخدام الحقل (Job) كحقل فهرسى لإنشاء ملف الفهرس (Job.ndx).

وفى السطر الثالث تم تجميع بيانات ملف قاعدة البيانات المفتوح وهو الملف (Employees) بناء على حقل المهنة (Job) ونتيجة لذلك تم إنشاء ملف قاعدة البيانات (Job.dbf).

وعند عرض بيانات حقلى المرتب (Salary) وساعات العمل (Hours) فى ملف قاعدة البيانات الجديد يتم ذلك كالآتى :

```
USE Job
LIST salary , hours
```

يلاحظ فى هذه الحالة ظهور الآتى مثلا :

Record	Job	salary	hours
1	Engineer	256570.00	700
1	Technical	189500.00	650

ويلاحظ هنا مجموع مرتبات المهندسين وكذلك الفنيين.

١٤٣ - الأمـــر (TYPE)

يستخدم هذا الأمر فى عرض محتويات ملف نصوص (Text File) والصورة العامة له كالآتى :

```
TYPE <filename>  [TO PRINT]
```

واسم الملف (Filename) يجب أن يتضمن الإمتداد ورمز وحدة الأقراص المخزن عليها إذا كانت غير وحدة الأقراص الحالية (Current drive).

١٤٤ - الأمـــر (UPDATE)

يستخدم هذا الأمر فى تحديث بيانات ملف قاعدة البيانات المفتوح من ملف آخر مفتوح فى منطقة عمل أخرى ويتم إدخال التعديلات بمطابقة الملفين تبعا لقيمة حقل معين يستخدم كمفتاح للمطابقة والصورة العامة لهذا الأمر كالآتى :

```
UPDATE ON <key field> FROM <alias>
 REPLACE <field1> WITH <exp1>
 [,<field2> WITH <exp2> ...][RANDOM]
```

حيث (key field) هو الحقل الفهرسى الذى يتم التحديث بناء عليه ويجب أن يكون الملفان مفهرسين (Indexed) أو مفروزين (Sorted) بناء على هذا الحقل وذلك لكى يقف المؤشر الخاص بكل ملف على نفس السجل فى الملفين مع كل حركة له. و (alias) هو الإسم المرادف الخاص بالملف الآخر المطلوب التحديث منه بالإضافة إلى منطقة العمل الخاصة به. ويجب ملاحظة أن الملف المطلوب تحديثه يكون مفتوحا وفى منطقة العمل التى تم اختيارها بواسطة الأمر (SELECT).

والإختيار (REPLACE) يستخدم فى إستبدال محتويات الحقول التى يراد تحديثها بمحتويات الحقول الموجودة فى الملف الآخر المستخدم فى التحديث. والإختيار(RANDOM) يستخدم عندما يكون ملف قاعدة البيانات المطلوب تحديثه مفهرسا (Indexed) على الحقل الفهرسى (Key Field) وليس مفروزا (Sorted).

مثال

عند تحديث بيانات الملف الرئيسى (Master) لقاعدة بيانات المخازن وذلك عن طريق ملف المبيعات (Sales) بناء على حقل رقم الجزء (Part_no) يتم كتابة الأوامر التالية :

```
SELECT 2
USE Sales
SELECT 1
USE MASTER INDEX MASTER
UPDATE ON Part_no FROM Sales ;
REPLACE Qty WITH Qty-Sales -> Qty
```

فى هذه الحالة يتم استبدال كمية المخزون (Qty) الموجودة فى الملف الرئيسى (Master) بنفس هذه القيمة (Qty) مطروحا منها قيمة المبيعات الموجودة فى حقل الكمية (Qty) الخاص بملف المبيعات (Sales).

١٤٥ - الأمـــر (USE)

يستخدم هذا الأمر فى فتح ملف قاعدة البيانات وملفات الفهرس المرتبطة به ، وإذا كان ملف قاعدة البيانات يحتوى على حقول ملاحظات (memo fields) يتم فتح ملف الملاحظات (dbt.) آليا. والصورة العامة لهذا الأمر كالآتى :

```
USE [<filename1>/?][INDEX <filename2>]
        [ALIAS <aliasname>]
```

والإسم (Filename1) هو إسم ملف قاعدة البيانات. ويمكن استخدام علامة الإستفهام (?) فى عرض ملفات قواعد البيانات المخزنة على القرص لاختيار الملف المطلوب منها.

والإسم (filename2) هو ملف الفهرس المرتبط بقاعدة البيانات المفتوحة. ويمكن فتح حتى سبعة ملفات فهرس مع ملف قاعدة بيانات واحد. ويمكن استخدام الأمر (USE) دون كتابة أى شىء بعده وهذا يؤدى إلى إغلاق جميع الملفات المفتوحة.

ويستخدم الإسم المرادف (alias name) فى تحديد إسم الملف متضمنا منطقة العمل المفتوحة.

١٤٦ - الأمر (WAIT)

يستخدم هذا الأمر فى إيقاف تنفيذ البرنامج والإنتظار حتى يضغط المستخدم على أى مفتاح والصورة العامة له كالآتى :

```
WAIT [<message>] [TO <memvar>]
```

حيث (message) هى رسالة يتم عرضها للمستخدم لتنبيهه إلى الضغط على أى مفتاح لاستمرار البرنامج.

والإختيار (TO <memvar>) يستخدم عندما يراد تخزين الحرف الذى يضغط عليه المستخدم فى متغير ذاكرة ، ويمكن كتابة الأمر (WAIT) دون كتابة أى شىء بعده وفى هذه الحالة يتم عرض الرسالة "..Press any key to continue" وهى الرسالة المبدئية (Default).

مثال

لإيقاف البرنامج مؤقتا وإعطاء المستخدم الإختيار ليستمر أو ينهى تشغيل البرنامج يتم كتابة السطور التالية :

```
WAIT 'Do you want to continue? (Y/N)' TO mcon
IF UPPER (mcon) # 'Y'
   RETURN
ENDIF
```

١٤٧ - الأمر (ZAP)

يستخدم هذا الأمر فى مسح جميع السجلات من ملف قاعدة البيانات المفتوح والصورة العامة له كالآتى :

```
ZAP
```

وهذا الأمر يماثل استخدام الأمر (DELETE ALL) وبعده الأمر (PACK). ويتبع ذلك مسح جميع ملفات الفهرس والملفات الأخرى المرتبطة بقاعدة البيانات.

الفصل التاسع والثلاثون

أهـم الـدوال المستخـدمـة



حتى يستطيع مخطط البرامج التحكم فى البرنامج وفى قاعدة البيانات يحتاج إلى الإلمام بمعظم الدوال (Functions) المستخدمة بواسطة برنامج (+DBaseIII) أو باقى برامج عائلة (DBase) مثل (DBaseIV) ، (+ FoxBase) ، (FoxPro). وفى هذا الفصل يتم شرح معظم هذه الدوال مع توضيح وظائفها بالأمثلة كلما أمكن.

ملاحظة

القيم الموجودة بين أقواس مربعة ([ ]) هى قيم إختيارية يستطيع المستخدم إدخالها أو عدم إدخالها حسب الحاجة.

١ - الدالــــــة (&)

تستخدم هذه الدالة فى التعويض بالماكرو (Macro Substitution) ويحدث ذلك عندما يراد التعويض عن قيمة متغير معين فى مكان يعامل فيه هذا المتغير كحروف (Characters). فمثلا عندما يراد البحث عن المتغير (mname) الذى يحتوى على الإسم (Mohamed) فإن كتابة الأمر (FIND mname) لاتؤدى إلى البحث عن الإسم (Mohamed) ولكنها تؤدى إلى البحث عن الحروف (mname). فى حين يمكن كتابة الأمر (FIND &mname) وفى هذه الحالة يتم البحث عن محتويات المتغير(mname) وهى (Mohamed). والصورة العامة لهذه الدالة كالآتى :

& <character variable> [.<exp>]

ويلاحظ أن المتغير فى هذه الحالة يجب أن يكون متغيرا حرفيا (Character Variable). ويستخدم الإختيار (<exp>.) عندما يراد إضافة حروف معينة فى نهاية المتغير الحرفى وفى هذه الحالة تستخدم النقطة (.) لتحديد نهاية المتغير الحرفى.

فمثلا لإدخال علامة الضرب (X) داخل سلسلة حرفية مع إضافة أرقام بعد علامة الضرب يمكن كتابة السطور التالية :

```
STORE "X" TO alpha
STORE "15 & alpha.30"TO show
```

```
?   show
15 X 30
```

ويلاحظ فى هذه الحالة ظهور العدد (15) ثم علامة الضرب (X) مكان المتغير (alpha) ثم العدد (30) كما كان مكتوبا فى السلسلة الحرفية.

وعندما يراد عرض رسالة معينة متضمنة إسم الشخص المطلوب عرض هذه الرسالة عليه يتم كتابة السطور التالية :

```
STORE "Mahmoud" TO mname
STORE "Hello & mname" TO greeting
?    greeting
```

ويلاحظ فى هذه الحالة ظهور الرسالة "Hello Mahmoud". وإذا تغيرت محتويات المتغير (mname) إلى أى إسم آخر فإن الرسالة تتضمن الإسم الجديد مثل (Hello Magdy) مثلا.

وإذا أراد مخطط البرامج إعطاء الفرصة للمستخدم لإدخال إسم ملف قاعدة البيانات المطلوب استخدامه فيمكنه مثلا إنشاء متغير ذاكرة لإسم الملف مثل (dname) ثم يقوم المستخدم بإدخال إسم الملف المطلوب فى هذا المتغير. وعن طريق دالة الماكرو (&) يمكن لمخطط البرامج فتح ملف قاعدة البيانات الذى أدخل المستخدم إسمه معها وذلك كالآتى :

```
USE & dname
```

٢ - دالـــة القيمة المطلقة (ABS)

تستخدم هذه الدالة فى الحصول على القيمة المطلقة لقيمة عددية والصورة العامة لها كالآتى :

```
ABS(<exp>)
```

حيث (exp) هى القيمة العددية المراد إيجاد القيمة المطلقة لها ويلاحظ أنها توضع بين قوسين.

وتستخدم هذه الدالة بصفة خاصة عندما يراد إيجاد الفرق العددى بين قيمتين دون الحاجة إلى معرفة أيهما أكبر من الأخرى كما يلاحظ من السطور التالية :

```
i = 20
J = 80
?   ABS(i-j)
```

وفى هذه الحالة يظهر الفرق (٦٠) موجبا.

٣ - الدالـــــة (ASC)

تستخدم هذه الدالة فى الحصول على شفرة الآسكى الخاصة بأول حرف من سلسلة حرفية معينة والصورة العامة لها كالآتى :

```
ASC (<exp>)
```

فمثلا عند كتابة السطر التالى :

```
ASC (Nagy)
```

يظهر العدد (78) الذى يمثل شفرة الآسكى الخاصة بالحرف (N). وتستخدم هذه الدالة بصفة خاصة فى قوائم الإختيارات عندما يراد اختبار الحرف الذى يدخله المستخدم لتنفيذ أحد الإختيارات.

٤ - الدالـــــة (AT)

تستخدم هذه الدالة فى البحث عن سلسلة حرفية فرعية (Substring) داخل سلسلة حرفية أخرى وعندما تجدها فإنها تعطى عددا يمثل ترتيب بداية هذه السلسلة الفرعية بالنسبة إلى بداية السلسلة الحرفية الأخرى. وإذا كانت السلسلة الحرفية التى يتم البحث عنها (Substring) غير موجودة داخل السلسلة الحرفية (String) فإن هذه الدالة تعطى القيمة صفر (0). والصورة العامة لهذه الدالة كالآتى :

```
AT(<exp1>, <exp2>)
```

حيث <exp1> هى السلسلة المطلوب البحث عنها داخل السلسلة الأكبر<exp2>.

مثال

يمكن كتابة السطر التالى :

```
?AT ("is" , "This is a test")
```

وفى هذه الحالة يلاحظ ظهور الرقم (3) وذلك لأن الحروف (is) تبدأ من الحرف الثالث (بالرغم من تكرارها إبتداء من الحرف السادس لأن الدالة (AT) تعطى رقم أول ظهور للسلسلة فقط).

٥ - الدالــــة (BOF)

تستخدم هذه الدالة لاختبار بداية الملف (Beginning of file) والصورة العامة لها كالآتى :

```
BOF()
```

وهى تعطى قيمة منطقية صحيح (True) أو غير صحيح (False) وتستخدم بصفة خاصة عندما يراد البحث عن سجل معين بفحص الملف عكسيا أى من نهاية الملف إلى بدايته. فمثلا يمكن كتابة السطور التالية :

```
USE Client
GO BOTTOM
DO WHILE.NOT.BOF()
    IF Job = "Teacher"
        ? RECNO ()
    ENDIF
    SKIP -1
ENDDO
```

٦ - الدالـــــة . (CDOW)

تستخدم هذه الدالة للحصول على إسم اليوم فى تاريخ معين والصورة العامة لها كالآتى :

CDOW(<exp>)

حيث (exp) قد يكون متغير ذاكرة تاريخى أو حقل تاريخى أو تاريخ اليوم الحالى. فمثلا إذا كان تاريخ اليوم الحالى (DATE()) هو (02/17/90) فلإيجاد إسم هذا اليوم يتم كتابة السطر التالى :

```
? CDOW (DATE())
```

وفى هذه الحالة يظهر الآتى :

```
Saturday
```

٧ - الدالـــــة (CHR)

وتستخدم هذه الدالة فى الحصول على الحروف والأعداد والحروف الخاصة عن طريق معرفة شفرة الآسكى الخاصة بهذه الحروف والصورة العامة لها كالآتى :

CHR(<exp>)

حيث (exp) هو رقم صحيح من (١) إلى (٢٥٥). وتستخدم هذه الدالة عندما يراد استخدام بعض الحروف الخاصة التى لايمكن كتابتها عن طريق لوحة المفاتيح فى رسم أشكال معينة على الشاشة وكذلك فى تشغيل الجرس (Bell) لتحذير المستخدم عند حدوث خطأ معين. فمثلا لتشغيل الجرس يتم كتابة السطر التالى :

```
? CHR(7)
```

ولإضافة رسالة تحذيرية بعد تشغيل الجرس يتم كتابة السطر التالى :

```
? CHR(7) + "Uncorrect....try again"
```

٨ - الدالـــــة (CMONTH)

تستخدم هذه الدالة فى الحصول على إسم الشهر فى تاريخ معين والصورة العامة لها كالآتى :

```
CMONTH(<exp>)
```

حيث (exp) هو متغير ذاكرة أو حقل تاريخى أو تاريخ اليوم الحالى.

مثـال

للحصول على إسم الشهر من تاريخ اليوم الحالى (02/17/90) يتم كتابة السطر التالى :

```
? CMONTH (DATE())
```

وفى هذه الحالة يظهر إسم الشهر (February).

ولتحديد إسم الشهر الذى يلى تاريخ اليوم الحالى بستين يوما مثلا يتم كتابة السطر التالى :

```
? CMONTH (DATE() +60)
```

وفى هذه الحالة يظهر إسم الشهر (April).

٩ - الدالـــــة (COL)

تستخدم هذه الدالة فى تحديد رقم العمود (Column) الذى يقف عنده المؤشر (Cursor). ويحدث ذلك عندما يراد التحكم فى مكان المؤشر على الشاشة وعلى الطابعة من خلال البرنامج والصورة العامة له كالآتى :

```
COL()
```

ويساعد ذلك على تحريك المؤشر إلى أماكن مختلفة بالنسبة للمكان الحالى للمؤشر فمثلا يمكن كتابة السطر التالى :

```
@1,COL() + 5 SAY "Enter your name"
```

وفى هذه الحالة يتم تحريك المؤشر خمسة أعمدة بعد آخر عمود كان يقف عنده.

١٠ - الدالـــة (CTOD)

تستخدم هذه الدالة فى تحويل التاريخ الحرفى من حروف إلى تاريخ والصورة العامة لها كالآتى :

```
CTOD(<exp>)
```

وتستخدم عندما يراد مقارنة تاريخ بتاريخ آخر أو لتحديد الفترة الزمنية المحصورة بين تاريخ وتاريخ آخر.

مثال

عندما يراد طباعة تقرير للبيانات الموجودة فى ملف قاعدة بيانات الطلبة وذلك بالنسبة للطلبة الملتحقين بالمعهد إبتداء من تاريخ معين يقوم المستخدم بإدخاله وحتى تاريخ معين يقوم بإدخاله أيضا فى هذه الحالة يتم كتابة السطور التالية :

```
STORE SPACE (8) TO start,end
@10,5 SAY " Enter start date " GET start
@12,5 SAY " Enter end date" GET end
READ
strat = CTOD (start)
end  = CTOD (end)
CLEAR
```

```
REPORT FORM Cadrep ;
FOR date_ent >= start .AND. date_ent <= end
```

ويلاحظ من السطور السابقة أنه تم تحويل التاريخ الذى يدخله المستخدم من حروف إلى تاريخ وذلك لاستخدامه فى تحديد بداية ونهاية البيانات التى تظهر فى التقرير.

١١ - الدالــــة (DATE)

تستخدم هذه الدالة فى الحصول على تاريخ اليوم الحالى والصورة العامة لها كالآتى :

```
DATE()
```

مع ملاحظة أن التاريخ يظهر على الصورة الأمريكية وهى (mm/dd/yy) إذا لم يتم تغيير صورة التاريخ بواسطة الأمر (SET DATE) أو الأمر (SET CENTURY) كما سبق الإيضاح.

مثــال

للحصول على تاريخ اليوم الحالى يتم كتابة السطر التالى :

```
? DATE()
```

وفى هذه الحالة يظهر التاريخ التالى مثلا : (02/20/90). ويمكن تخزين تاريخ اليوم الحالى فى متغير ذاكرة لاستخدامه بعد ذلك فى البرنامج وذلك كالآتى :

```
STORE DATE () TO mdate
```

١٢ - الدالــــة (DAY)

تستخدم هذه الدالة فى الحصول على العدد الممثل لترتيب اليوم فى الشهر بالنسبة لتاريخ معين والصورة العامة لها كالآتى :

```
DAY(<exp>)
```

حيث (exp) هو متغير ذاكرة تاريخى أو حقل تاريخى أو تاريخ اليوم الحالى مثلا.

مثـال

للحصول على ترتيب اليوم الحالى وهو (02/17/90) بالنسبة للشهر يتم كتابة السطر التالى :

```
? DAY (DATE())
```

فى هذه الحالة يظهر الرقم (17) الذى يمثل ترتيب اليوم فى الشهر.

١٣ - الدالـــــة (DBF)

تستخدم هذه الدالة فى الحصول على إسم ملف قاعدة البيانات المفتوح فى منطقة العمل التى تم اختيارها بواسطة الأمر (SELECT) والصورة العامة لهذه الدالة كالآتى :

```
DBF()
```

وفى حالة عدم وجود أى ملف قاعدة بيانات مفتوح تعطى هذه الدالة سلسلة حرفية خالية (Null String). وتستخدم هذه الدالة عندما يراد معرفة إذا كان هناك ملف قاعدة بيانات (DBF) مفتوح قبل بداية البرنامج أم لا. وفى حالة وجود ملف مفتوح يتم إغلاقه ثم إعادة فتحه فى نهاية البرنامج حتى تعود حالة البرنامج إلى وضعها الأصلى . ولتنفيذ ذلك يتم كتابة السطور التالية :

```
Null = ""
IF   NULL < DBF()
    old_file =DBF()
    USE Clients
    -------
    -------
    -------  Commands
    -------
```

```
--------
    USE & old_file
ENDIF
```

وفى هذا البرنامج يتم اختبار الشرط الموجود بعد (IF) للتأكد من وجود ملف قاعدة بيانات مفتوح قبل البرنامج. فإذا كان هناك ملف مفتوح يتم تخزين إسمه فى متغير الذاكرة (old_file) وذلك حتى يتسنى فتحه بعد ذلك ثم يتم إغلاقه عن طريق فتح ملف الموظفين (Clients) وتنفيذ الأوامر المطلوبة على هذا الملف. وقبل نهاية البرنامج يتم فتح الملف الذى كان مفتوحا مرة ثانية. ويلاحظ هنا استخدام دالة التعويض (&) لأن إسم الملف (DBF) ليس معروفا ولكنه مخزن فى متغير الذاكرة (old_file).

١٤ - الدالـــــة (DELETED)

تستخدم هذه الدالة فى تحديد السجلات التى تم وضع علامات عليها تمهيدا لمسحها. والصورة العامة لها كالآتى :

```
DELETED()
```

وهذه الدالة تعطى القيمة صحيح (True) إذا كان السجل الحالى تم وضع علامة عليه لمسحه (Marked for deletion) ويفيد ذلك عندما يريد المستخدم استعراض السجلات التى تم وضع علامات عليها قبل مسحها نهائيا حتى يتأكد أنها السجلات المطلوب مسحها. فمثلا لعرض بيانات السجلات التى تم تجهيزها للمسح يمكن كتابة السطر التالى :

```
DISPLAY FOR DELETED
```

١٥ - الدالـــــة (DISKSPACE)

تستخدم هذه الدالة فى تحديد حجم الذاكرة المتاح على القرص والصورة العامة لها كالآتى :

```
DISKSPACE()
```

وهى تعطى عددا صحيحا يمثل عدد الحروف (Bytes) المتاحة على القرص. وتفيد عندما يراد عمل نسخ إحتياطية من ملف قاعدة البيانات وكذلك عندما يراد عمل فرز للسجلات (Sorting) حيث أن الفرز يتطلب إنشاء ملف قاعدة بيانات جديد بالإضافة إلى الملف الأصلى.

فمثلا إذا كان المتغير (mfilesize) يحتوى على عدد الحروف (Bytes) التى يتكون منها ملف قاعدة البيانات فيمكن كتابة السطور التالية لإجراء عملية الفرز(Sorting) :

```
IF DISKSPACE() > mfilesize * 2
   SORT ON Last_name TO Newfile
ELSE
   @ 20,15 SAY "There is not enough space"
ENDIF
```

١٦ - الدالـــــة (DOW)

وتستخدم هذه الدالة فى الحصول على رقم يمثل ترتيب اليوم فى الأسبوع والصورة العامة لها كالآتى :

```
DOW (<exp>)
```

حيث (exp) هو متغير ذاكرة تاريخى أو حقل تاريخى أو تاريخ اليوم الحالى مع ملاحظة أن الرقم (1) يمثل يوم الأحد (Sunday). فمثلا للحصول على اليوم الممثل للتاريخ الحالى يتم كتابة السطر التالى :

```
? DOW (DATE())
```

فى هذه الحالة يظهر الرقم (7) الممثل ليوم السبت.

١٧ - الدالـــــة (DTOC)

تستخدم هذه الدالة فى تحويل التاريخ إلى حروف والصورة العامة لها كالآتى :

DTOC (<EXP>)

حيث (exp) هو متغير ذاكرة تاريخى أو حقل تاريخى أو التاريخ الحالى. وتفيد هذه الدالة عندما يراد عرض الحقول التاريخية على الشاشة على صورة التاريخ المعروفة.

١٨ - الدالـــــة (EOF)

تستخدم هذه الدالة فى تحديد نهاية ملف قاعدة البيانات (End Of File) والصورة العامة لها كالآتى :

EOF()

وهى تعطى القيمة المنطقية صحيح (True) عندما يصل مؤشر السجلات إلى آخر الملف. وهذا لايعنى أن المؤشر يكون عند آخر سجل فى الملف ولكنه يتخطى هذا السجل ويصل إلى علامة نهاية الملف. وتستخدم هذه الدالة عندما يراد تنفيذ حلقة تكرارية على جميع سجلات قاعدة البيانات.

مثـال

عندما يراد مثلا عرض بيانات الموظفين الذين تزيد أعمارهم عن (٤٠) سنة يتم كتابة الحلقة التكرارية التالية :

```
DO WHILE.NOT.EOF()
  LIST FOR age > 40
ENDDO
```

١٩ - الدالـــــة (ERROR)

تستخدم هذه الدالة فى الحصول على رقم يحدد الخطأ الذى قد يحدث أثناء تنفيذ البرنامج والصورة العامة لها كالآتى :

ERROR()

وتستخدم بصفة خاصة عندما يراد علاج الأخطاء التى قد تحدث أثناء تنفيذ البرنامج وذلك عن طريق إغلاق بعض الملفات أو تغيير الأقراص أو مسح بعض الملفات لتوفير مساحة تخزينية وهكذا.

مثـــال

يمكن إدخال السطر التالى فى برنامج لإكتشاف أى أخطاء قد تحدث به.

```
ON ERROR DO Err_prg WITH ERROR()
```

فعندما يحدث أى خطأ (Error()) يتم تنفيذ البرنامج الفرعى (Err_prg) مع إدخال الرقم الممثل لهذا الخطأ (ERROR) كمعامل (Parameter) للبرنامج الفرعى. أما البرنامج الفرعى فى هذه الحالة فيكون كالآتى مثلا :

```
PARAMETERS Error_no
   IF Error_no =54
   -----
   ----- commands
   -----
   ENDIF
```

حيث الأوامر (Commands) فى هذه الحالة تكون رسائل للمستخدم لاتخاذ بعض الاجراءات للتخلص من هذا الخطأ.

٢٠ - الدالـــــة (FIELD)

تستخدم هذه الدالة للحصول على إسم أى حقل بمعلومية ترتيب هذا الحقل بين حقول قاعدة البيانات والصورة العامة لها كالآتى :

```
FIELD (<exp>)
```

حيث (exp) هو العدد الذى يمثل ترتيب الحقل وهو ينحصر بين (١) و (١٢٨).

مثـال

للحصول على عدد حقول قاعدة البيانات من خلال البرنامج يتم كتابة السطور التالية :

```
USE Clients
num_fields = 0
null= ""
DO WHILE null < FIELD (num_fields + 1)
   num_fields = num_fields + 1
ENDDO
? num_fields
```

وفى هذا البرنامج يتم تنفيذ الحلقة التكرارية طالما كان إسم الحقل الذى يتم الحصول عليه بواسطة الدالة (FIELD (num_fields +1)) أكبر من السلسلة الحرفية الخالية (null string) وهذا يعنى أن الحلقة التكرارية تستمر طالما كان هناك حقول داخل الملف وفى هذه الحالة يتم زيادة عدد الحقول واحدا. وهكذا يتم تحديد عدد الحقول بعد إنتهاء تنفيذ الحلقة التكرارية.

٢١ - الدالـــــة (FOUND)

هذه الدالة تعطى القيمة (صحيح) أى (True) عندما يصل البرنامج إلى السجل الذى يتم البحث عنه بواسطة الأمر (FIND) أو الأمر (SEEK) أو الأمر (LOCATE) أو الأمر (CONTINUE). والصورة العامة لها كالآتى :

```
FOUND()
```

وتستخدم هذه الدالة فى البرنامج عندما يراد تنفيذ بعض الإجراءات فى حالة الوصول إلى السجل المطلوب أو عدم الوصول إليه.

مثـال

لاستخدام هذه الدالة مع الأمر (LOCATE) يمكن كتابة السطور التالية :

```
LOCATE FOR Job = "Teacher"
```

```
DO WHILE FOUND()
    ? name,address
    CONTINUE
ENDDO
```

وفى هذه الحالة يذهب المؤشر إلى أول سجل يحقق الشرط فإذا وجد أول سجل يتم تنفيذ الحلقة التكرارية التى يتم عن طريقها عرض بيانات حقول الإسم والعنوان الخاصة بهذا السجل ثم يتم البحث عن السجل التالى عن طريق الأمر (CONTINUE). وهكذا يستمر تنفيذ الحلقة التكرارية طالما كان السجل موجودا فى كل مرة.

ويمكن استخدام هذه الدالة مع الأمر (SEEK) كالآتى مثلا :

```
SEEK "Teacher"
IF FOUND()
    DO WHILE Job = "Teacher"
      ? name, address
      SKIP
    ENDDO
ENDIF
```

ويراعى فى هذه الحالة أن يكون الملف مفهرسا على حقل الوظيفة (Job).

٢٢ - الدالـــــة (IIF)

تستخدم هذه الدالة لإدخال جملة (IF) الشرطية على سطر واحد بدلا من إدخالها على عدة سطور. والصورة العامة لها كالآتى :

IIF(<exp1>,<exp2>,<exp3>)

حيث (exp1) هو الشرط المراد اختباره فإذا تحقق فإن الدالة تعطى القيمة (exp2) وإذا لم يتحقق تعطى القيمة (exp3). وهى تسمى (IF) السريعة حيث أنها تؤدى إلى سرعة تنفيذ البرنامج وزيادة كفاءته.

مثال

يمكن ملاحظة الفرق بين استخدام الأمر (IF-ENDIF) واستخدام الدالة (IIF) من خلال هذا المثال حيث يتم كتابة السطور التالية التى توضح استخدام الأمر (IF-ENDIF).

```
IF sex = "F"
   mname = "Ms." + name
ESLE
   mname = "Mr." + name
ENDIF
```

وتنفيذ هذه السطور يؤدى إلى تخزين الإسم الموجود فى حقل الإسم (name) فى متغير الذاكرة (mname) مسبوقا بالحروف (.Ms) إذا كان السجل خاصا بأنثى (Female) أى أن السجل يحتوى على القيمة (F) فى حقل الجنس (Sex). كما يخزنه مسبوقا بالحروف (.Mr) إذا كان السجل خاصا بذكر فى الأحوال الأخرى أى السجلات التى تحتوى على أى قيمة أخرى غير (F). وإذا أريد استخدام الدالة (IIF) لتنفيذ نفس العملية يتم كتابة السطر التالى :

```
mname = IIF(sex = "F" , "Ms." , "Mr.") + name
```

وعند تنفيذ هذا السطر يتم اختبار الشرط (Sex = "F") فإذا تحقق يتم إضافة الحروف (.Ms) قبل الإسم الموجود فى الحقل (name) وتخزين القيمة الناتجة فى المتغير (mname). وإذا لم يتحقق يتم إضافة الحروف (.Mr) قبل الإسم الموجود فى الحقل (name) وتخزين القيمة الناتجة فى المتغير (mname).

٢٣ - الدالـــــة (INKEY)

هذه الدالة تعطى القيمة العددية الممثلة لآخر حرف تم الضغط عليه بواسطة المستخدم. والصورة العامة لها كالآتى :

```
INKEY()
```

وهى تعطى عددا صحيحا بين (صفر) و (٢٥٥) يقابل شفرة الآسكى (ASCII Code) الخاصة بهذا الحرف وتستخدم عندما يراد اختبار الحروف التى يضغط عليها المستخدم.

مثال

السطور التالية توضح استخدام عداد للوقت يحدد الزمن الذى يقضيه المستخدم قبل إدخال الإختيار المطلوب.

```
DO WHILE.T.
-----
-----
----- Menu Options
-----
-----
i = 0
DO WHILE i = 0
   @ 1,72 SAY TIME()

i = INKEY()
ENDDO
DO CASE
   CASE CHR(i)$ "Aa"
     DO <program1>
   CASE CHR(i)$ "Bb"
     DO <program2>
   CASE CHR(i)$ "Cc"
     DO <program3>
   CASE CHR(i)$ "Qq"
     RETURN
ENDCASE
ENDDO
```

ويؤدى تنفيذ الحلقة التكرارية الداخلية إلى عرض الوقت عن طريق الدالة (TIME()) طالما كانت (i=0) حيث (i) تمثل القيمة العددية للحرف الذى يضغط عليه المستخدم، لذلك يتم حساب الوقت المستهلك حتى يضغط المستخدم على أى حرف يمثل أحد الإختيارات الموجودة فى القائمة.

٢٤ - الدالـــــة (INT)

تستخدم هذه الدالة فى تحويل القيم العددية إلى أعداد صحيحة عن طريق حذف أى كسور عشرية والصورة العامة لها كالآتى :

INT(<exp>)

مثـال

لتحويل العدد (10.23) الى عدد صحيح يتم كتابة السطر التالى :

```
? INT(10.23)
```

ويلاحظ ظهور العدد (10) فى هذه الحالة.

٢٥ - الدالة (ISALPHA)

تستخدم هذه الدالة فى اختبار أول حرف فى قيمة معينة فاذا كان حرفا هجائيا فانها تعطى القيمة صحيح أى (True) واذا كان رقما أو حرفا من الحروف الخاصة فإنها تعطى القيمة غير صحيح أى (False) والصورة العامة لهذه الدالة كالآتى :

ISALPHA(<exp>)

حيث (exp) هو القيمة التى يتم اختبارها.

مثـال

يمكن كتابة السطر التالى :

```
? ISALPHA("abc123")
```

وفى هذه الحالة تظهر القيمة (.T.) أى صحيح وذلك لأن أول حرف هو الحرف (a). كما يمكن كتابة السطر التالى :

```
? ISALPHA("123abc")
```

وفى هذه الحالة تظهر القيمة (.F.) أى غير صحيح وذلك لأن أول حرف ليس حرفا هجائيا.

٢٦ - الدالة (ISCOLOR)

تعطى هذه الدالة القيمة صحيح أى (True) اذا كان البرنامج يعمل على حالة الألوان (Color Mode). وتعطى القيمة غير صحيح أى (False) إذا كان يعمل على حالة اللون الأحادى (Monochrome). والصورة العامة لها كالآتى :

```
ISCOLOR()
```

وتفيد هذه الدالة فى إعطاء مخطط البرامج إمكانية التحكم فى تصميم البرنامج بجعله يعمل على الشاشة الملونة أو الأحادية اللون حسب بيئة الحاسب المتوفرة.

مثال

للتحكم فى الألوان من خلال البرنامج أى استخدام الألوان فى حالة الشاشة الملونة (Color Mode) واستخدام الأبيض والأسود فى حالة الشاشة الأحادية اللون يتم كتابة السطور التالية :

```
IF ISCOLOR()
   SET Color TO GR/B,W/R,GR
ELSE
   SET COLOR TO W+
ENDIF
```

٢٧ - الدالة (ISLOWER)

تعطى هذه الدالة القيمة صحيح (True) عندما تبدأ القيمة الحرفية التى يتم اختبارها بحرف صغير (Lowercase) كما تعطى القيمة غير صحيح (False) عندما تبدأ بحرف كبير (Uppercase) والصورة العامة لها كالآتى :

```
ISLOWER()
```

مثـال

لاختبار السلسلة الحرفية (abc123) يتم كتابة السطر التالى :

```
? ISLOWER ( "abc123" )
.T.
```

ويلاحظ هنا ظهور القيمة المنطقية (.T.) أى صحيح.

ولاختبار السلسلة الحرفية (ABC123) يتم كتابة السطر التالى :

```
? ISLOWER ("ABC123")
.F.
```

ويلاحظ هنا ظهور القيمة المنطقية (.F.) أى غير صحيح.

٢٨ - الدالة (ISUPPER)

وهى عكس الدالة السابقة أى أنها تعطى القيمة صحيح أى (True) عندما تبدأ القيمة الحرفية التى يتم اختبارها بحرف كبير (Uppercase) كما تعطى القيمة غير صحيح أو (False) عندما تبدأ بحرف صغير (Lowercase) والصورة العامة لها كالآتى :

```
ISUPPER ()
```

٢٩ - الدالـة (LEFT)

هذه الدالة تعطى عددا من حروف السلسلة الحرفية بدءا من اليسار والصورة العامة لها كالآتى :

LEFT (<exp1>,<exp2>)

حيث (exp1) هو السلسلة الحرفية المراد سحب جزء منها.

و (exp2) هو عدد يمثل عدد الحروف المطلوب استخراجها من اليسار.

وهذه الدالة تشبه الدالة (SUBSTR) مع اختلاف واحد وهو أنها لاتحتاج الى تحديد بداية السلسلة الحرفية المستخرجة حيث أنها تبدأ دائما من أول حرف من اليسار.

مثال

للحصول على الثلاثة حروف الأولى من الإسم (Mohamed) يتم كتابة السطر التالى:

```
? LEFT ("Mohamed",3)
Moh
```

يلاحظ ظهور الحروف (Moh) .

٣٠ - الدالـة (LEN)

هذه الدالة تعطى عددا يمثل عدد الحروف الموجودة فى سلسلة حرفيـة (String). والصورة العامة لها كالآتى :

LEN(<exp>)

حيث (<exp>) هو السلسلة الحرفية المراد حساب طولها.

مثـال

لإيجاد طول الحقل (name) الذى يحتوى على الاسم (Hatem Zaky) يتم كتابة السطر التالى :

```
? LEN (name)
25
```

يلاحظ فى هذه الحالة ظهور الرقم (25) مع أن الإسم المذكور يحتوى على عشرة حروف فقط. وذلك لأن الدالة (LEN) تحسب طول الحقل بالكامل متضمنا الفراغات (Spaces).

ولحساب الطول الفعلى للإسم يتم التخلص من الفراغات (Spaces) الموجودة بعد الإسم باستخدام الدالة (TRIM) ثم حساب طول السلسلة الحرفية بعد ذلك وذلك كالآتى :

```
? LEN (TRIM(name))
```

ويلاحظ فى هذه الحالة ظهور العدد (10) الممثل للعدد الفعلى للحروف متضمنا المسافة (Space) الموجودة بين الإسمين فقط حيث ان الدالة (TRIM) قد أزالت المسافات الموجودة آخر الإسم .

٣١ - الدالـة (LOG)

هذه الدالة تعطى قيمة اللوغاريتم الطبيعى لأى عدد والصورة العامة لها كالآتى :

```
LOG (<exp>)
```

حيث (<exp>) هو العدد المطلوب إيجاد اللوغاريتم الطبيعى له. واللوغاريتم الطبيعى هو الذى يكون أساسه النسبة التقريبية (e). فمثلا لإيجاد اللوغاريتم للعدد (2.718) الذى يمثل النسبة التقريبية (e) يتم كتابة السطر التالى :

```
? LOG (2.718)
1.00
```

٣٢ - الدالـة (LOWER)

تستخدم هذه الدالة فى تحويل الحروف الكبيرة إلى حروف صغيرة والصورة العامة لها كالآتى :

```
LOWER(<exp>)
```

حيث (<exp>) هو السلسلة الحرفية المطلوب تحويلها إلى حروف صغيرة، وتستخدم هذه الدالة بصفة خاصة فى الحالات التى يراد فيها التحكم فى البيانات التى يدخلها المستخدم فمثلا اذا كان المطلوب من المستخدم إدخال بيانات الإسم (name) بحيث تكون بحروف صغيرة حتى تكون كل السجلات متماثلة يتم استخدام هذه الدالة فى تحويل البيانات التى يدخلها المستخدم إلى حروف صغيرة (Lowercase). فإذا أدخل المستخدم حروفا صغيرة أو كبيرة يتم تحويلها إلى صغيرة.

مثال

عندما يراد البحث عن إسم معين فى قاعدة بيانات الطلبة مثلا يتم كتابة السطور التالية :

```
@ 15,15 SAY "Enter name to look for" ;
GET Lookup
READ
Lookup = LOWER (Lookup)
SEEK Lookup
```

يلاحظ فى هذه الحالة تحويل الإسم الذى يدخله المستخدم إلى حروف صغيرة قبل البحث عنه باستخدام الأمر (SEEK) وذلك بفرض أن الأسماء قد سبق تخزينها فى قاعدة البيانات بحروف صغيرة.

٣٣ - الدالة (LTRIM)

تستخدم هذه الدالة فى مسح المسافات الخالية (Spaces) من أول السلسلة الحرفية من اليسار والصورة العامة لها كالآتى :

LTRIM(<exp>)

وتفيد هذه الدالة عندما يراد علاج الأخطاء التى قد تنتج عن إدخال المستخدم لمسافات خالية قبل البيانات التى يقوم بإدخالها حيث يتم مسح هذه المسافات قبل إدخالها إلى الحقول.

فمثلا لكى يدخل المستخدم إسما معينا فى حقل الإسم (name) يتم أولا إنشاء متغير ذاكرة لهذا الحقل مثل (mname). ثم يتم استخدام الدالة السابقة فى التخلص من أى مسافات سواء فى أول الإسم أو فى آخره وذلك كالآتى مثلا :

```
mname = space(30)
@ 10,10 SAY "Enter a name" GET mname
READ
mname = LOWER (LTRIM(TRIM(mname)))
REPLACE name WITH mname
```

فى السطر الأول يتم إنشاء متغير الذاكرة (mname).

وفى السطر الثانى يتم عرض رسالة للمستخدم لإدخال الإسم.

وفى السطر الثالث يتم تخزين الإسم فى المتغير (mname).

وفى السطر الرابع يتم تحويل الإسم إلى حروف صغيرة مع التخلص من المسافات فى أول الإسم وآخره باستخدام الدالتين (LTRIM) و (TRIM).

وفى السطر الخامس يتم استبدال محتويات حقل الإسم للسجل الحالى بالإسم الموجود فى متغير الذاكرة (mname).

٣٤ - الدالة (LUPDATE)

هذه الدالة تعطى تاريخ آخر تحديث تم اجراؤه للملف والصورة العامة لها كالآتى :

LUPDATE()

وتتيح هذه الدالة لمخطط البرامج التحكم فى تحديث المستخدم للبيانات حتى لايتم تحديثها عدة مرات حيث أن تحديثها عدة مرات قد يؤدى إلى إدخال سجلات مكررة أو تجميع بيانات عددية أكثر من مرة مما يؤدى فى النهاية الى عدم دقة البيانات.

مثال

يمكن عن طريق السطور التالية عرض رسالة للمستخدم توضح له آخر تاريخ تم فيه تحديث الملف. ثم تترك له حرية الإختيار بين تحديث البيانات إذا كانت هناك بيانات جديدة مطلوب تحديثها بعد هذا التاريخ أو الإكتفاء بالتحديث الذى سبق إجراؤه.

```
IF LUPDATE() < DATE()
   entry = "?"
   @ 5,5 SAY "last entry was on"+DTOC(LUPDATE());+
      "Enter now ?(Y/N)" GET entry PICTURE "Y"
   READ
   IF entry = "y"
         DO <entry program>
   ENDIF
ENDIF
```

ويلاحظ فى السطر الثالث استخدام علامة (+) ثم الفاصلة المنقوطة (;) وذلك لربط السلسلة الحرفية فى هذا السطر بالسلسلة الحرفية المكملة لها فى السطر التالى. وعند الرغبة فى تحديث بيانات الملف يتم إدخال (Y) فيتم تنفيذ البرنامج (entry program) الذى يؤدى إلى إدخال البيانات المطلوب تحديثها.

٣٥ - الدالـــــة (MAX)

هذه الدالة تعطى أكبر قيمة من قيمتين عدديتين والصورة العامة لها كالآتى :

```
MAX(<exp1>, <exp2>)
```

٣٦ - الدالـــــة (MIN)

هذه الدالة عكس الدالة السابقة حيث تعطى أقل قيمة من قيمتين عدديتين والصورة العامة لها كالآتى :

```
MIN(<exp1>,<exp2>)
```

وتستخدم هذه الدالة عندما يراد مثلا الحصول على أقل قيمة عددية لحقل معين. فمثلا للحصول على أقل مرتب لموظف فى قاعدة بيانات الموظفين (Clients) يتم كتابة السطور التالية :

```
USE Clients
min_sal = salary
DO WHILE.NOT.EOF()
   min_sal = MIN(min_sal,salary)
   SKIP
ENDDO

USE
RETURN
```

فى هذا البرنامج يتم الإنتقال إلى السجل التالى دائما بواسطة الأمر (SKIP) ثم يتم الحصول على أقل قيمة من المرتب (Salary) والمرتب السابق ثم يتم تخزين هذه القيمة فى المتغير (min_sal) وهكذا تتكرر هذه العملية حتى يتم الحصول على أقل قيمة للمرتب.

ملاحظة

يمكن استخدام نفس الطريقة فى الحصول على أكبر قيمة بواسطة الدالة (MAX).

٣٧ - الدالـــة (MOD)

هذه الدالة تعطى باقى القسمة الصحيحة لعدد على عدد آخر. والصورة العامة لها كالآتى :

```
MOD(<exp1>, <exp2>)
```

وتستخدم هذه الدالة بصفة خاصة فى التحويل من وحدات إلى أخرى مثل تحويل الياردة إلى بوصة والمتر إلى سنتيمتر والساعة إلى دقائق وثوان وهكذا. فمثلا إذا كان هناك عدد من الدقائق يراد تحويله إلى عدد من الأيام وعدد من الساعات وعدد من الدقائق يمكن كتابة السطور التالية :

```
t = 36500
minutes = MOD(t,60)
h = INT(t/60)
hours = MOD(h,24)
days = INT(h/24)
? t,"minutes are:",days,"days",hours,"hours";
  minutes, "minutes"
```

وعند الضغط على مفتاح الإدخال بعد السطر الأخير يلاحظ ظهور الآتى :

```
36500 minutes are:25 days 8 hours 20 minutes
```

والسطر الأول من البرنامج يتم عن طريقه إنشاء المتغير (t) الذى يحتوى على العدد (36500) دقيقة.

والسطر الثانى يقوم بتحديد عدد الدقائق التى تبقى بعد القسمة على (٦٠) وهى الدقائق التى تبقى بعد تحديد الأيام والساعات.

والسطر الثالث يقوم بتحديد عدد الساعات الصحيحة الموجودة الناتجة عن قسمة عدد الدقائق على (٦٠).

والسطر الرابع يقوم بتحديد الساعات الباقية بعد القسمة على (٢٤).

والسطر الخامس يحدد عدد الأيام الصحيحة الناتجة عن قسمة الساعات على(٢٤).

والسطر السادس يقوم بعرض البيانات التى تم حسابها على الشاشة.

٣٨ - الدالــــة (MONTH)

هذه الدالة تعطى عددا يمثل ترتيب الشهر فى السنة بالنسبة لتاريخ معين والصورة العامة لها كالآتى :

```
MONTH(<exp>)
```

حيث (exp) هو متغير ذاكرة حرفى أو حقل حرفى أو تاريخ اليوم الحالى.

فمثلا إذا كان تاريخ اليوم الحالى (2/18/1990) يمكن الحصول على رقم الشهر كالآتى :

```
? MONTH(DATE())
```

فى هذه الحالة يظهر الرقم (2) الممثل لترتيب الشهر.

٣٩ - الدالــــة (NDX)

هذه الدالة تعطى إسم ملف الفهرس المفتوح فى منطقة العمل (Work Area) التى سبق اختيارها بواسطة الأمر (SELECT) والصورة العامة لهذه الدالة كالآتى :

```
NDX(<exp>)
```

حيث (exp) هو رقم يمثل ترتيب ملف الفهرس بين الملفات المفتوحة وهو يأخذ أى رقم من (1) إلى (7).

مثـال

لكى يعرف البرنامج أسماء ملفات الفهرس المفتوحة ويتعامل معها يمكن كتابة السطور التالية :

```
i =1
Null= ""
DO WHILE Null < NDX(i).AND.i <= 7
   ? NDX(i)
   i =i +1
ENDDO
```

٤٠ - الدالــــة (OS)

هذه الدالة تعطى إسم نظام التشغيل الذى يعمل عليه البرنامج والصورة العامة لها كالآتى :

```
OS()
```

وتستخدم هذه الدالة عندما يراد تصميم برنامج نقال (Portable) أى يمكنه العمل على نظم تشغيل مختلفة مثل (MS-DOS) و (UNIX).

مثـال

لتحويل البرنامج الذى يعمل على نظام التشغيل (MS-DOS) ليعمل على نظام التشغيل (UNIX) يتم كتابة السطور التالية :

```
STORE OS() TO opsys
IF SUBSTR(opsys,1,4) = "UNIX"
   DO setunix
ENDIF
```

وتؤدى هذه السطور إلى اختبار نظام التشغيل المستخدم فإذا كان النظام (UNIX) يتم تنفيذ البرنامج (setunix) الذى يؤدى إلى تجهيز البرنامج للتعامل مع نظام التشغيل (UNIX).

٤١ - الدالــــة (PCOL)

تستخدم هذه الدالة فى تحديد العمود (Column) الذى يقف عنده رأس الطباعة بالنسبة للورقة الموجودة على الطابعة والصورة العامة لها كالآتى :

```
PCOL()
```

ويمكن عن طريق هذه الدالة تحريك رأس الطباعة على الورقة فى أعمدة مختلفة بالنسبة للعمود الذى يقف عنده. فمثلا عندما يراد الطباعة بعد آخر طباعة سبق تنفيذها بخمسة أعمدة يتم كتابة السطور التالية :

```
SET DEVICE TO PRINT
@ 1,PCOL()+ 5 SAY "This is a test"
SET DEVICE TO SCREEN
```

كما يمكن عن طريق هذه الدالة معرفة رقم العمود الذى يقف عنده رأس الطباعة بالنسبة للورقة كالآتى مثلا :

```
? PCOL()
```

فى هذه الحالة يظهر العدد (5) مثلا أى أن رأس الطباعة يقف عند العمود (5) من الورقة وهكذا.

٤٢ - الدالــــة (PROW)

تستخدم هذه الدالة فى تحديد السطر (Row) الذى يقف عنده رأس الطباعة بالنسبة للورقة والصورة العامة لها كالآتى :

```
PROW()
```

كما تستخدم أيضا فى تحريك رأس الطباعة عددا من السطور بالنسبة لآخر سطر كان يقف عنده. فمثلا عندما يراد الطباعة بعد آخر طباعة سبق تنفيذها بخمسة سطور (PROW()+5) يتم كتابة السطور التالية :

```
SET DEVICE TO PRINT
@ PROW()+ 5,1 SAY "This is a test"
SET DEVICE TO SCREEN
```

٤٣ - الدالـــــة (RECCOUNT)

هذه الدالة تعطى عدد السجلات (Records) فى ملف قاعدة البيانات المفتوح. والصورة العامة لها كالآتى :

```
RECCOUNT()
```

وتستخدم هذه الدالة بصفة خاصة فى البرامج التى يتم من خلالها عمل نسخ إحتياطية (Backups) للفات قواعد البيانات آليا. حيث يتم استخدامها مع الدالة (RECSIZE()) والدالة (DISKSPACE()) فى تحديد الحجم المتاح على القرص وإذا كان يكفى لعمل نسخة إحتياطية للملف أم لا.

مثـال

لإيجاد عدد السجلات فى ملف بيانات الطلبة يتم كتابة السطور التالية :

```
USE cadets
? RECCOUNT
```

فى هذه الحالة يظهر العدد الذى يمثل عدد السجلات فى الملف.

٤٤ - الدالـــــة (RECNO)

تستخدم هذه الدالة فى تحديد رقم السجل (Record number) الذى يقف عنده المؤشر والصورة العامة لها كالآتى :

RECNO()

وتفيد هذه الدالة بصفة خاصة بعد أوامر البحث مثل الأمر (SEEK) لمعرفة إذا كان هناك سجل يحقق الشرط أم لا. فمثلا عندما يراد البحث عن إسم معين موجود فى المتغير (Lookup) يتم كتابة السطور التالية :

```
SEEK Lookup
Recno = RECNO()
IF Recno>0
    SET FORMAT TO Cadets
    READ
    CLOSE FORMAT
ELSE
    @ 10,10 SAY "There is no & lookup"
    ? CHR(7)
ENDIF
```

وفى هذا البرنامج يتم البحث عن الإسم الموجود فى متغير الذاكرة (Lookup) فإذا كان موجودا يتم تخزين رقم هذا السجل فى متغير الذاكرة (Recno) وإذا لم يكن موجودا يتم تخزين القيمة صفر فى هذ المتغير. وفى الحالة الأولى يتم فتح شاشة الإدخال عن طريق الأمر (SET FORMAT) ليقوم المستخدم بإدخال البيانات المطلوبة وفى الحالة الثانية يتم عرض الرسالة المبينة. ويلاحظ أهمية استخدام الماكرو فى هذه الرسالة لإظهار محتويات المتغير (Lookup) للمستخدم ليعرف أن هذا الإسم غير موجود.

٤٥ - الدالـــــة (RECSIZE)

هذه الدالة تعطى حجم السجل (Record Size) فى ملف قاعدة البيانات المفتوح. والصورة العامة لها كالآتى :

RECSIZE()

وتستخدم هذه الدالة بصفة خاصة فى البرامج التطبيقية التى يتم من خلالها عمل

نسخ إحتياطية (Backups) للفات قواعد البيانات المستخدمة وذلك بالإشتراك مع الدالة (RECCOUNT()) والدالة (DISKSPACE()) حيث تساعد على التأكد من وجود مساحة متاحة فى القرص لتخزين النسخة الإحتياطية. فمثلا لعمل نسخة من ملف قاعدة بيانات كبير يحتاج إلى عدة أقراص لتخزينه يتم كتابة السطور التالية :

```
USE File1
SET DEFAULT TO B
DO WHILE .NOT. EOF()
  WAIT "Insert new disk in drive B, and press a key."
  COPY NEXT(DISKSPACE() - <header size>)/RECSIZE() ;
  TO Backup
  SKIP
ENDDO
USE
```

حيث (header Size) هو حجم العنوان الذى يكون موجودا قبل كل سجل ويتم حسابه من العلاقة الآتية :

```
header size =32*<number of fields> + 35
```

والبرنامج السابق يؤدى إلى استمرار نسخ السجلات سجلا سجلا طالما كانت القيمة الناتجة بعد الأمر (COPY NEXT) أكبر من واحد. وعندما تقل هذه القيمة عن واحد فإن هذا يعنى أن المساحة الخالية (Disk Space) المتاحة على القرص أقل من حجم السجل التالى وبالتالى يتوقف النسخ ويطلب البرنامج من المستخدم وضع قرص جديد ثم تتكرر هذه العملية حتى يتم نسخ ملف قاعدة البيانات.

٤٦ - الدالــــة (REPLICATE)

تستخدم هذه الدالة فى تكرار حرف معين أو قيمة حرفية معينة عددا من المرات يتم تحديده. والصورة العامة لهذه الدالة كالآتى :

```
REPLICATE(<exp1>,<exp2>)
```

حيث (exp1) هو القيمة الحرفية المراد تكرارها. و(exp2) هو العدد الذى يمثل عدد مرات تكرار هذه القيمة.

ويجب ملاحظة أن عدد الحروف الذى يتكون من عملية التكرار يجب ألا يزيد عن (٢٥٤) حرفا. وتستخدم هذه الدالة بصفة خاصة فى تكوين أشكال على الشاشة مثل المستطيلات التى يتم تكوين أضلاعها من حروف معينة مثل الحرف (*) أو عن طريق استخدام أى حروف أخرى يتم تكوينها باستخدام الدالة (CHR()).

فمثلا للحصول على خط أفقى مكون من تكرار الحرف (*) عددا محددا من المرات يتم كتابة السطر التالى :

```
5,5 SAY REPLICATE("*",20)
```

وفى هذه الحالة يظهر الآتى على الشاشة :

```
* * * * * * * * * * * * * * * * * * * *
```

٤٧ - الدالـــــة (RIGHT)

هذه الدالة تعطى عددا من الحروف الموجودة فى سلسلة حرفية بدءا من اليمين والصورة العامة لها كالآتى :

```
RIGHT(<exp1> , <exp2>)
```

حيث (exp1) هو القيمة الحرفية المراد استخراج عدد من حروفها.

و (exp2) هو العدد الذى يمثل عدد الحروف المراد استخراجه. وتستخدم هذه الدالة فى حالات كثيرة يراد فيها استخراج جزء من سلسلة حرفية معينة.

مثـال

لاستخراج الحروف الثلاثة الأخيرة من الإسم (Mahmoud) يتم كتابة السطر التالى :

```
? RIGHT("Mahmoud",3)
```

فى هذه الحالة يلاحظ ظهور الحروف الثلاثة (oud).

٤٨ - الدالـــة (ROUND)

هذه الدالة تقوم بتقريب العدد لعدد محدد من الكسور العشرية والصورة العامة لها كالآتى :

```
ROUND(<exp1> , <exp2>)
```

حيث (exp1) هو العدد المطلوب تقريبة. و(exp2) هو عدد الكسور العشرية المطلوب التقريب إليها.

مثـال

لتقريب العدد (10.765788) لأقرب رقمين عشريين يتم كتابة السطر الآتى :

```
? ROUND(10.765788 ,2)
```

وعند الضغط على مفتاح الإدخال يظهر العدد (10.77).

٤٩ - الدالـــة (ROW)

هذه الدالة تعطى السطر الحالى الذى يقف عنده المؤشر على الشاشة والصورة العامة لها كالآتى :

```
ROW()
```

وتستخدم عندما يراد التحكم فى مكان المؤشر وتحريكه عددا من السطور بالنسبة للمكان الذى يقف عنده فمثلا يمكن كتابة السطر التالى :

```
@ ROW()+5,3 SAY "Enter your name"
```

فى هذه الحالة تظهر الرسالة المبينة بعد خمسة سطور من آخر سطر وصل إليه المؤشر.

٥٠ - الدالـــــة (RTRIM)

تستخدم هذه الدالة فى مسح المسافات من نهاية قيمة حرفية معينة والصورة العامة لها كالآتى :

```
RTRIM(<exp>)
```

وهى تماثل الدالة (TRIM()) تماما.

٥١ - الدالـــــة (SPACE)

تستخدم هذه الدالة فى إنشاء متغير ذاكرة يحتوى على عدد معين من الحروف الخالية (blanks) والصورة العامة لها كالآتى :

```
SPACE(<exp>)
```

ويمكن تكوين متغير ذاكرة يحتوى على عدد من الحروف يصل إلى (٢٥٤) حرفا.

مثـال

عندما يراد مثلا إنشاء متغير ذاكرة (mname) حتى يستطيع المستخدم إدخال الإسم فيه يتم كتابة السطور التالية :

```
mname = SPACE(30)
@5,5 SAY "Enter new name" GET mname
READ
```

٥٢ - الدالـــــة (SQRT)

هذه الدالة تعطى الجذر التربيعى للقيمة العددية الموجبة والصورة العامة لها كالآتى :

```
SQRT(<exp>)
```

مثال

لإيجاد الجذر التربيعى للعدد (4) يتم كتابة السطر التالى :

```
? SQRT(4)
```

فى هذه الحالة يظهر العدد (2.00).

٥٣ - الدالــــة (STR)

تستخدم هذه الدالة فى تحويل القيمة العددية إلى سلسلة حرفية (String) والصورة العامة لها كالآتى :

```
STR(<exp>, <length> , <decimal>)
```

حيث (exp) هو القيمة العددية المراد تحويلها.

و (length) هو عدد الأرقام المراد ظهورها وهو اختيارى وفى حالة عدم إدخاله يظهر حتى عشرة أرقام.

و (decimal) هو عدد الأرقام العشرية وهو اختيارى أيضا وفى حالة عدم إدخاله يتم التقريب لأقرب عدد صحيح.

وعند إدخال طول (length) أصغر من عدد الأرقام الصحيحة الموجودة فى العدد فإن البرنامج يعرض مجموعة من حروف النجمة (*) مكان العدد. وعند إدخال عدد أرقام عشرية (Decimal) أقل من عدد الأرقام العشرية الموجود فى العدد يتم تقريب الأرقام العشرية الزائدة.

مثال

لعرض العدد (33.56) كحروف (String) يتم كتابة السطر التالى :

```
? STR(33.56 ,4,1)
```

فى هذه الحالة يظهر العدد (33.6).

ويلاحظ هنا عند تحديد الطول (Length) تم حساب نقطة الكسر العشرى (Decimal Point) ضمن عدد الأرقام فأصبح العدد (4) بدلا من (3).

٥٤ - الدالـــة (STUFF)

تستخدم هذه الدالة فى تغيير أى جزء داخل سلسلة حرفية معينة والصورة العامة لها كالآتى :

```
STUFF(<exp1>,<start position>, ;
<number of characters>,<exp2>)
```

حيث (exp1) هى السلسلة الحرفية المطلوب التعديل فيها.

و (exp2) هى السلسلة الحرفية المطلوب ادخالها فى السلسلة الحرفية الأولى.

و (start position) هى قيمة عددية تمثل المكان المطلوب إدخال السلسلة الحرفية بدءا منه.

و (number of characters) هو عدد الحروف المطلوب استبدالها من السلسلة الحرفية الأولى وإذا كان هذا العدد صفرا يتم إدخال السلسلة الحرفية الثانية داخل السلسلة الحرفية الأولى دون تغيير الحروف الموجودة فى السلسلة الأولى. أى يتم حشر السلسلة الثانية داخل السلسلة الأولى (Insertion). وإذا كانت السلسلة الثانية عبارة عن سلسلة خالية (null string) يتم مسح حروف من السلسلة الأولى بقدر طول السلسلة الخالية.

مثال

إذا أريد تغيير عنوان معين داخل قاعدة بيانات الطلبة (Cadets) تكتب السطور التالية :

```
. new_street = nasr city
. USE Cadets
. GO 5
. ? address
10 -Ainshams -Cairo
. REPLACE address WITH STUFF(address,4,9,new_street)
. ?address
```

يلاحظ فى هذه الحالة ظهور العنوان (10-nasr city-Cairo) بدلا من العنوان السابق.

٥٥ - الدالـــــة (SUBSTR)

تستخدم هذه الدالة فى استخراج جزء من سلسلة حرفية معينة والصورة العامة لها كالآتى :

SUBSTR(<exp1>, <starting position>
,[<number of characters>])

حيث (exp1) هو السلسلة الحرفية المطلوب استخراج جزء منها.

و (starting position) هو المكان الذى يبدأ منه استخراج السلسلة الحرفية الفرعية.

و (number of characters) هو عدد إختيارى يمثل عدد الحروف المراد سحبها من السلسلة الحرفية. وفى حالة عدم كتابة هذا العدد يتم استخراج السلسلة الحرفية بدءا من مكان البداية (starting position) إلى آخر السلسلة الحرفية.

مثال

فى المثال السابق الخاص بالعنوان الموجود فى السجل الخامس يراد معرفة المدينة التى يقع فيها هذا العنوان. لتنفيذ ذلك يتم كتابة السطر التالى :

```
?SUBSTR(address,14,5)
```

فى هذه الحالة يظهر الآتى :

```
Cairo
```

٥٦ - الدالـــة (TIME)

هذه الدالة تعطى الوقت الحالى الذى تم إدخاله عند بدء تشغيل الجهاز من خلال نظام التشغيل والصورة العامة لها كالآتى :

```
TIME()
```

مثـال

للحصول على الوقت الحالى يتم كتابة السطر التالى :

```
?TIME()
```

وعند الضغط على مفتاح الإدخال يظهر الآتى مثلا :

```
20:45:20
```

٥٧ - الدالـــة (TRANSFORM)

تستخدم هذه الدالة للتحكم فى شكل المخرجات التى تنتج من الأوامر (?,??,DISPLAY,LABEL,LIST,REPORT) والصورة العامة لها كالآتى :

```
TRANSFORM(<exp1> , <exp2>)
```

وهى تؤدى نفس العمل الذى تؤديه عبارة (PICTURE) مع الأمر (@...SAY).

مثال

لعرض الأسماء الموجودة فى الحقل (name) بحيث يتم فصل كل حرف عن الحرف التالى بمسافة (Space) يتم كتابة السطر التالى :

```
DISPLAY TRANSFORM(name,@Rxxxxxxxx)
```

فإذا كان الإسم الموجود في هذا الحقل هو (Mohamed) مثلا يظهر الآتى :

```
M o h a m e d
```

ولعرض مرتب أربعة موظفين بحيث يتكون المرتب من ثمانية أرقام مع رقمين عشريين يتم كتابة السطر التالى :

```
LIST NEXT 4 TRAN$FORM(salary, "# # # #. # #")
```

فى هذه الحالة تظهر الأعداد كالآتى :

Record #	Salary
1	570.50
2	600.80
3	1000.00
4	700.00

٥٨ - الدالـــــة (TRIM)

تستخدم هذه الدالة فى مسح المسافات الخالية من نهاية السلسلة الحرفية والصورة العامة لها كالآتى :

```
TRIM(<exp>)
```

وهى تفيد عندما يراد التأكد من التخزين الصحيح للبيانات التى يدخلها المستخدم حيث يتم أولا إدخال هذه البيانات فى متغير ذاكرة بعد التخلص من المسافات الخالية فى

أوله أو فى آخره ثم يتم نقل هذه البيانات من متغير الذاكرة إلى الحقل الخاص بها. فمثلا عندما يراد إدخال أى إسم فى الحقل (name) يتم كتابة الأمر التالى لإنشاء متغير الذاكرة (mname).

```
STORE SPACE(30) TO mname
```

ثم يتم التخلص من المسافات الخالية فى أول الإسم وآخره كالآتى :

```
STORE LTRIM(TRIM(mname)) TO mname
```

وهذا يؤدى إلى إدخال الحروف التى يدخلها المستخدم دون أى مسافات قبلها أو بعدها.

وتستخدم هذه الدالة أيضا عندما يراد معرفة الطول الصحيح لأى سلسلة حرفية باستخدام الدالة (LEN) وذلك كالآتى مثلا :

```
LEN(LTRIM(TRIM(mname)))
```

٥٩ - الدالـــــة (TYPE)

تستخدم هذه الدالة فى تحديد نوع أى قيمة أو متغير معين والصورة العامة لها كالآتى :

```
TYPE(<exp>)
```

وهى تعطى حرفا كبيرا (Capital) يمثل نوع القيمة (exp) مثل (C) للقيم الحرفية (Characters) ، (N) للقيم العددية (Numeric) ، (L) للقيم المنطقية (Logical) ، (M) لحقول الملاحظات (Memo) ، (U) للقيم غير المعرفة (Undefined).

مثـال

عندما يراد اختبار المتغير (score) مثلا يتم كتابة السطر التالى :

```
? TYPE("score")
```

فى هذه الحالة يظهر الحرف (U) وهذا يعنى أن المتغير غير معرف (Undefined) وذلك لأن المتغير لم يتم تعريفه قبل هذا الأمر. أما عند كتابة الآتى مثلا :

```
STORE 100 TO score
TYPE("score")
```

فى هذه الحالة يظهر الحرف (N) وهذا يعنى أن المتغير عددى.

٦٠ - الدالـــــة (UPPER)

تستخدم هذه الدالة فى تحويل الحروف الصغيرة (Lowercase) إلى حروف كبيرة (Upercase) والصورة العامة لها كالآتى :

```
UPPER(<exp>)
```

وتفيد هذه الدالة فى التأكد من إدخال البيانات التى يدخلها المستخدم بنفس شكل البيانات المخزنة فى الملف كما تفيد أيضا عندما يتم عرض رسالة على المستخدم واستقبال الرد على هذه الرسالة والتعامل مع هذا الرد بصرف النظر عن إدخاله بحروف كبيرة أو صغيرة.

مثـال

عندما يراد البحث عن إسم معين فى حقل الإسم (name) يتم أولا إنشاء متغير ذاكرة (Lookup) مثلا لاستقبال الإسم الذى يدخله المستخدم ثم تحويل هذا المتغير إلى حروف كبيرة حتى يماثل الحروف الموجودة فى حقل الإسم لجميع السجلات ثم يتم البحث عن هذا الإسم باستخدام الأمر (SEEK). ولتنفيذ ذلك يتم كتابة السطور التالية :

```
Lookup = SPACE(15)
@10,10 SAY "Enter name of person to edit" GET Lookup
READ
Lookup = UPPER(Lookup)
SEEK Lookup
```

٦١ - الدالـــة (VAL)

تستخدم هذه الدالة فى تحويل الأرقام الموجودة فى السلسلة الحرفية (String) إلى العدد المقابل والصورة العامة لها كالآتى :

VAL(<exp>)

وهى تقوم بعكس العمل الذى تؤديه الدالة (STR) مع ملاحظة أن البيانات الموجودة فى القيمة (exp) يجب أن تكون أعدادا وليست حروفا.

مثـال

يمكن كتابة السطور التالية :

```
STORE "88.50" TO string
VAL(string)
```

فى هذه الحالة يظهر نفس العدد (88.50) ولكن الفرق هنا أن هذه القيمة عددية أى يمكن التعامل معها بالجمع عليها أو الطرح منها وهكذا.

٦٢ - الدالـــة (VERSION)

هذه الدالة تعطى رقم نسخة برنامج (+DBaseIII) أو أى برنامج آخر من برامج عائلة (DBase) المستخدمة مثل (DBaseIV) ، (FoxBase +) ، (FoxPro).
والصورة العامة لها كالآتى :

VERSION()

وتستخدم فى البرامج التى تتطلب بعض الخصائص المرتبطة بنسخة البرنامج المستخدمة.

٦٣ - الدالـــــة (YEAR)

تستخدم هذه الدالة فى الحصول على العدد الممثل للسنة داخل تاريخ معين والصورة العامة لها كالآتى :

```
YEAR(<exp>)
```

حيث (exp) هو متغير ذاكرة تاريخى أو حقل تاريخى أو تاريخ اليوم الحالى.

مثـال

إذا كان تاريخ اليوم الحالى هو (02/18/93) فيمكن الحصول على العدد الممثل للسنة كالآتى :

```
?YEAR(DATE())
```

يلاحظ فى هذه الحالة ظهور العدد (1993).





الجزء الرابع

تحليل وتصميم النظم

وأدوات الـ (CASE)



الفصل الأربعون

مقدمة



في العهود الماضية لتحليل النظم وكتابة البرامج كانت هناك قواعد محدودة يتم تطبيقها على هذا المجال. وعندما تطورت أجهزة الحاسب وأصبحت أكبر قدرة وأكثر إنتشارا أصبح على المحللين والمصيمن تطوير أساليب العمل والإنتقال من مرحلة الهواية إلى مرحلة الإحتراف المعتمدة على التخطيط السليم. وأدى ذلك إلى ظهور الوسائل التركيبية أو البنائية (Structured Techniques) في مجال تحليل النظم والبرمجة. وتمتاز هذه الوسائل التركيبية بأنها ليست ساكنة (Static) وإنما هي في تطور مستمر إلى الأفضل وسوف تشهد الأيام القادمة تطورات ضخمة في هذه الوسائل بما سوف يؤدي إلى نظم آلية تتميز بالسرعة والكفاءة.

والأهداف الرئيسية للوسائل التركيبية تتلخص في الآتي :

* تصميم برامج عالية الكفاءة.
* تصميم برامج يمكن صيانتها أو تعديلها بسهولة.
* تبسيط وتسهيل عملية تطوير البرامج.
* تحقيق أكبر قدر من السيطرة والتحكم في عملية التطوير.
* تقليل زمن عملية تطوير النظام.
* تقليل تكلفة تطوير النظام.

ولتحقيق هذه الأهداف الرئيسية هناك مجموعة من الأهداف الفرعية يمكن تلخيصها في الآتي :

* تحليل المسائل أو التراكيب المعقدة إلى أخرى أقل تعقيدا والإستمرار في عملية التحليل للوصول إلى وظائف صغيرة. وهذا يؤدي إلى تفتيت التصميم إلى تسلسل هرمي من الروتينات الفرعية.
* تحقيق سهولة التصميم عن طريق وضوح العلاقات (Interfaces) بين الروتينات الفرعية.
* استخدام الجداول (Diagrams) في توضيح المشاكل المعقدة.
* تحسين قابلية القراءة (Readability) للجداول والبرامج. وهذا يتيح صيانة النظام وتطويره بواسطة شخص آخر غير الأشخاص القائمين بتصميمه.
* تحسين الإتصال مع المستخدم.
* تحقيق وحدة البناء.
* استخدام وسائل دقيقة يسهل تعلمها والتدريب عليها.

* استخدام مجموعة من تراكيب التحكم (Control Structures) يسهل تحويلها إلى شفرة بلغة من لغات البرمجة كما سوف يتم الإيضاح فى الفصول التالية.
* تحقيق سهولة الإتصال بين أفراد فريق التطوير (Development Team).
* تقليل عدد المبرمجين إلى الحد الأدنى (يمكن الوصول إلى فريق من مبرمج واحد).
* تقليل الأخطاء.
* إكتشاف الأخطاء مبكرا.
* إنشاء مكتبات تحتوى على روتينات قوية ومختبرة تصلح لبناء نظم أخرى.
* تحقيق إدارة جيدة للبيانات.
* إمداد المبرمج والمحلل بطاولة أدوات يستطيع بواسطتها الإستفادة من إمكانيات الحاسب فى تحقيق أهداف النظام.
* تحقيق أكبر قدر من آلية التصميم بمايتيح الوصول إلى التوليد الآلى للكود (Code Generation).

٤٠ - ١ تطور الوسائل التركيبية

ظهرت الوسائل التركيبية فى المجتمع الأكاديمى فى أواخر الستينات وأصبحت معروفة وشائعة فى بداية السبعينات. ومنذ هذا الوقت أكتسبت شهرة واسعة وأصبح لها أثر ملحوظ على فن البرمجة. وزادت أهمية هذه الوسائل مع ظهور أدوات هندسة البرامج (CASE Tools) وهى إختصار (Computer - Aided Software Engineering) حيث أصبحت ضرورية لتوفير الوسائل الآلية لتصميم النظام وتحقيقه (Verfication) وتوليد الكود آليا. والشكل (٤٠ - ١) يوضح مراحل تطور الوسائل التركيبية والفترات المختلفة التى تم خلالها هذا التطور كما يتم شرح هذه المراحل فى الأجزاء التالية.

٤٠ - ١ - ١ البرمجة التركيبية (Structured Programming)

ركزت الوسائل التركيبية فى البداية على هيئة البرنامج وكيف يمكن جعله أكثر وضوحا وكيف يمكن السيطرة على تعقيد البرنامج عندما يزيد حجمه بدرجة كبيرة. وكان ذلك يتم عن طريق إتخاذ بعض الإجراءات القياسية عند كتابة البرامج وسيتم إيضاح هذه الإجراءات فيما بعد.

Early 1970s — **STRUCTURED PROGRAMMING**
Structured Coding Conventions
Top-Down Programming
Parnas, Information Hiding
Dijkstra, Levels of Abstraction
Wirth, Stepwise Refinement



Mid 1970s — **STRUCTURED DESIGN**
Yourdon/Constantine, Structured Design
Jackson, Design Methodology
Warnier-Orr, Design Methodology



Late 1970s — **STRUCTURED ANALYSIS**
Demarco, Structured Analysis
Gane and Sarson, Structured Analysis
SADT
Requirements Design Language

DATA BASE TECHNIQUES

Codd, Third Normal Form
Data Modeling



Early 1980s — **AUTOMATED TECHNIQUES**
HOS, Axiomatic Verification
Automated Data Modeling
Intelligent Data Models
Nonprocedural Languages
Action Diagrams



Late 1980s — **CASE TECHNIQUES**
Martin, Information Engineering
Graphic Workbenches for System Analysts
Action Diagram Editors for Fourth-Generation Languages
Rule-Based Systems
Design Aids with Verification Checkin
Specifications from which Code is Generated Automatically

The evolution of structured techniques into CASE technology

شكل (٤٠ - ١)

٤٠ - ١ - ٢ التصميم التركيبى (Structured Design)

بدءا من منتصف السبعينات إمتدت الفلسفة التركيبية إلى مرحلة التصميم. وركزت هذه الفلسفة على العلاقة بين المشكلة المطلوب حلها وبين البرامج والروتينات المستخدمة فى الحل وظهر مبدأ تجزئة البرامج (Modularization) مع تحديد العلاقات بين الأجزاء المختلفة.

٤٠ - ١ - ٣ التحليل التركيبى (Structured Analysis)

عندما أصبح واضحا أن كثيرا من المشاكل تنتج عن سوء تحديد المتطلبات ، تحول الإهتمام إلى مرحلة التحليل. وظهرت الوسائل التركيبية فى التحليل فى أواخر السبعينات حيث إنتشر استخدام خرائط التدفق (Data Flow Diagrams) كما ظهرت وسائل تصميم قواعد البيانات وقاموس البيانات ... الخ.

وقد ظهر العديد من الوسائل التركيبية التى تمثلت فى منهجيات (Methodologies) وسياسات (Strategies) وأدوات (Tools) وفرت فى مجموعها أسلوبا منظما (Systematic Approach) لتطوير النظام. ورغم أن هذه الوسائل أختلفت فى وظائفها وفى مواقع استخدامها داخل دورة حياة النظام وفى نوع الأشخاص أو الكيانات المستخدمة لها ، إلا أنها جميعا تشترك فى الفلسفة التركيبية.

٤٠ - ١ - ٤ الوسائل الآلية (Automated Techiques)

مع بداية الثمانينات وصلت الإنتاجية (Productivity) فى البرمجة إلى أدنى معدلاتها فى الوقت الذى إنتشرت فيه أجهزة الحاسب وخاصة الحاسبات الصغيرة وانخفضت أسعارها بدرجة كبيرة. كما زاد عدد مستخدمى الحاسب وزادت مطالبهم من البرامج التطبيقية التى تغطى مختلف المجالات. وأصبحت المنهجيات التى تستخدم الأساليب السابقة والموضحة فى المستطيلات الثلاثة الأولى من الشكل (٤٠ - ١) تؤدى إلى بطء شديد فى تطوير النظم وإلى مشاكل متعددة فى الصيانة. وأدى السعى إلى زيادة الإنتاجية إلى ظهور لغات برمجة جديدة ، مولدات تقارير (Report Generations) ، مولدات تطبيقات (Application Generators) ، أدوات تصميم قواعد البيانات ، برمجيات خاصة بدعم إتخاذ القرار ، أدوات للمستخدم النهائى بالإضافة إلى مختلف وسائل تحديد المتطلبات التى تؤدى فى النهاية

إلى توليد الكود آليا.

ومنذ منتصف الثمانينات كانت هناك حاجة ملحة إلى مستويات أعلى من ميكنة عملية تطوير النظم. وأصبح معلوما لدى الكافة أن توظيف الحاسب فى عملية التطوير سوف يؤدى إلى ظهور وسائل وأدوات أكثر تطورا. وبدأ الحاسب يدخل فى عمليات إنشاء وتصحيح وتعديل الجداول والمخططات التركيبية المستخدمة فى توصيف النظام. كما استخدم الحاسب أيضا فى إنشاء قواميس البيانات (Data Dictionaies) والموسوعات (Encyclopedias) التى وفرت دعما كبيرا للمحللين والمصممين. كما استخدم الحاسب أيضا فى ميكنة عمليات تمثيل البيانات (Data Modeling) وإسترجاع مجموعات فرعية من نماذج البيانات لكل مصمم حسب حاجته. كما استخدم الحاسب أيضا فى تحقيق النظام (Verification) من خلال وسائل رياضية (Mathematical) فى بعض الأحيان. وفوق كل ذلك فقد استخدمت برمجيات الحاسب فى التغلب على البطء والأخطاء المصاحبين لعمليات تشفير البرامج وذلك من خلال توليد الكود آليا (Code Gereation).

٤٠ - ١ - ٥ أدوات الـ (CASE)

فى أوائل السبعينات ظهرت برامج التصميم باستخدام الحاسب (Computer Aided Design) والتى تختصر (CAD). وكان المحللون والمبرمجون - رغم ميكنتهم لمعظم أجزاء النظام - يعانون من عدم ميكنة وظائفهم. وعدم الإستفادة من الحاسب فى هذه الميكنة. وكانوا يستخدمون القلم والأدوات الهندسية فى إنشاء التصميمات التى لم تكن تخلو من الأخطاء والأجزاء الغير مطابقة أو الغير موجودة فى متطلبات المستخدم. لذلك ظهر ماعرف بالتحليل باستخدام الحاسب (Computer Aided Systems Analysis) ويختصر (CASA) وكانت الوسائل التركيبية هى حجر الزاوية فى هذه النظم. وتعتمد هذه النظم على استخدام الخصائص الجرافيكية للحاسب فى تصميم المخططات والتراكيب مع استخدام موسوعة مخزنة فى الحساب تحتوى على معلومات عن أجزاء التصميم المختلفة مع التأكد من شمولها (Consistency). وهذا يتيح إنشاء نظم معقدة مع القدرة على صيانة هذه النظم وتعديلها وتطويرها.

٤٠ - ٢ خصائص هامة

هناك خصائص هامة يجب توافرها فى الوسائل التركيبية من بينها ما يأتى :

* يتم تصميم هذه الوسائل بحيث توفر سهولة الاستخدام (User Friendly) وذلك بهدف إشراك المستخدم فى عمليات تحقيق النظام (Validation) ومناقشة التصميم ومدى مطابقته للمتطلبات.

* يتم تصميم هذه الوسائل بحيث يسهل تنفيذها بواسطة لغات الجيل الرابع (Fourth-gereration Languages) أو مولدات الكود(Code Generators).

* يتم تصميم هذه الوسائل بحيث تكون دقيقة جدا (Rigorous) وفى بعض الأحيان يمكن اختبارها بالوسائل الرياضية خاصة فى النظم الكبيرة.

* يجب أن تكون هذه الوسائل موجهة للاستخدام من خلال قواعد البيانات (Data Oriented).

* يتم تصميم هذه الوسائل للعمل مع وسائل التصميم الآلية مثل أدوات هندسة البرامج (CASE Tools) التى تكون أسرع وأدق وأكثر قابلية للتعديل (Changeable) من الوسائل التقليدية.

* يجب تعديل هذه الوسائل باستمرار حتى يمكن تحقيق أكبر إستفادة من استخدام الحاسب فيها. وهذه الخاصية سوف تؤدى على المدى الطويل إلى تغيير شامل فى طبيعة برمجة وتصميم النظم.

والفصول التالية من هذا الجزء توضح عمليات تحليل وتصميم النظم واستخدام الوسائل التركيبية فيها وكذلك استخدام أدوات الـ (CASE).

الفصل الحادي والأربعون

تحليل متطلبات النظام



أول خطوة فى تصميم النظام هى تحديد وتحليل متطلبات النظام. وهذه المتطلبات قد تكون وظيفية (Functional) أو غير وظيفية (Nonfunctional) وقد تكون ساكنة (Static) أو متحركة (Dynamic). وفى هذا الفصل يتم توضيح خصائص متطلبات النظام ووسائل تحديد هذه المتطلبات سواء كانت يدوية (Manual) أو آلية (Automatic). كما يتم توضيح وسائل توثيق هذه المتطلبات ومراجعتها.

عندما يطلب العميل بناء نظام معلومات فإن هذا العميل يعلم ما يجب أن يحققه هذا النظام. وفى الغالب يكون النظام موجودا والمطلوب هو استبداله بنظام جديد أى أن النظام يكون له هدف محدد. والمتطلبات (Requirements) هى أهداف هذا النظام والأشياء التى يستطيع تحقيقها لبلوغ هذه الأهداف. فمثلا نفرض أن المطلوب إنشاء نظام إصدار شيكات دفع (Paychecks) فإن أحد المتطلبات قد يكون إصدار الشيكات كل أسبوعين أو قد يكون السماح بتشغيل النظام من عدة مواقع فى المؤسسة. كل هذه المتطلبات هى توصيف محدد للعمليات والوظائف التى يحققها النظام وتؤدى إلى الوصول إلى الهدف العام له.

ويلاحظ أن المتطلبات السابق توضيحها لا تتضمن كيفية تنفيذ هذه المتطلبات. وبمعنى آخر ليس هناك أى تحديد لنوع قاعدة البيانات التى سوف تستخدم أو كمية الذاكرة المطلوب استخدامها فى الحاسب أو نوع اللغة المطلوب استخدامها فى تصميم النظام. أى أن المتطلبات توضح الهدف من النظام دون الخوض فى تفاصيل تحقيق النظام أو تصميمه. وهذا يعنى أن أى جزء من المتطلبات لا يكون مرتبطا بصورة أو بأخرى بالهدف يجب إلغاؤه من المتطلبات.

٤١ - ١ تحديد المتطلبات (Requirement Specification)

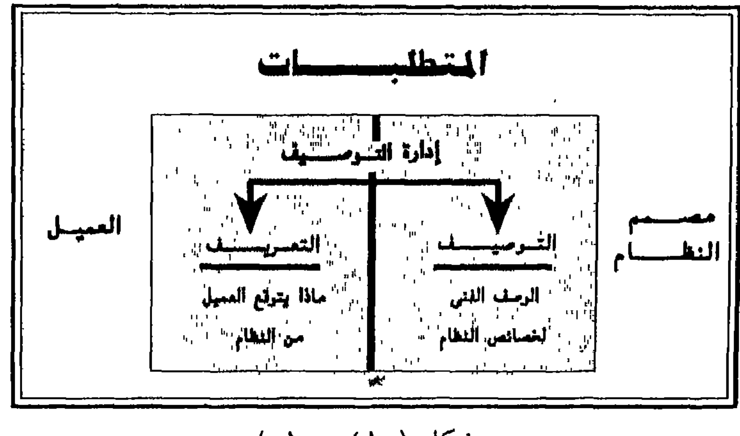
يمكن تصنيف متطلبات النظام حسب نوع التحليل الذى يتم إجراؤه عليها ونوع التوثيق الناتج عن هذا التحليل. والنوع الأول من التحليل ينتج ما يسمى بوثائق تعريف المتطلبات (Requirements Definition Documents). وتكون هذه الوثائق مكتوبة بالطريقة التى يفهمها العميل وهى تسجل كل شىء يتوقع العميل تحقيقه بواسطة النظام. وهذه الوثائق لاتحتوى على أى تفاصيل فنية يستطيع مصممو النظام الإعتماد عليها فى التصميم. فمثلا هذه الوثائق قد تحتوى على الآتى :

(يجب الحصول على معلومات الطلبة الراسبين بسرعة)

بينما يحتاج مصممو النظام إلى توصيف أكثر دقة كالآتى مثلا :

(يجب استرجاع سجلات الطلبة الراسبين خلال خمس ثوان).

وهذه الطريقة الأخيرة فى التوصيف تسمى وثائق توصيف المتطلبات (Requirement Specification Documents). وهذه الوثائق يتم فيها إعادة صياغة وثائق تعريف المتطلبات بالصورة التى تكون ملائمة لمصممى النظام. لذلك يجب أن يكون هناك تحويل مستمر لكل جزء من وثائق تعريف المتطلبات إلى الجزء المقابل فى وثائق توصيف المتطلبات. وهذا التحويل يسمى إدارة التوصيف(Configuration Management) وهو لايقتصر على مرحلة تحديد متطلبات النظام ولكنه يربط بين جميع مراحل تطوير النظام. أنظر شكل (٤١ - ١)



شكل (٤١ - ١)

٤١ - ٢ أنواع المتطلبات

توضح وثائق تعريف المتطلبات كل شىء عن علاقة النظام بالبيئة المحيطة (Environment). وتتضمن هذه الوثائق الآتى :

١ - البيئة الفعلية (Physical Environment)

* أين الأجهزة التى يتم استخدامها فى النظام ؟
* هل هناك موقع واحد أو عدة مواقع ؟

* هل هناك قيود بيئية مثل درجة الحرارة والرطوبة والتأثير المغناطيسى ؟

٢ - أدوات الإتصال (Interfaces)

* هل يتم الحصول على المدخلات من نظم أخرى ؟
* هل يتم توجيه المخرجات إلى نظم أخرى ؟
* هل هناك طريقة محددة لتشكيل البيانات ؟
* هل هناك وسائط محددة لتخزين البيانات ؟

٣ - المستخدمون (Users)

* من الذى سوف يستخدم النظام ؟
* هل سيكون هناك أنواع مختلفة من المستخدمين ؟
* ما هو المستوى المهارى لكل نوع من المستخدمين ؟
* ما هو مستوى التدريب المطلوب لكل نوع من المستخدمين ؟

٤ - الوظيفية (Functionality)

* ماذا سيحقق النظام ؟
* متى يقوم النظام بتحقيق وظائفه ؟
* كيف ومتى يتم تعديل أو تحسين النظام ؟
* هل هناك قيود على سرعة التنفيذ وسرعة الإستجابة ؟

٥ - التوثيق (Documentation)

* ماهو حجم التوثيق المطلوب ؟
* لمن سيتم تجهيز هذه الوثائق ؟

٦ - البيانات (Data)

* ماهو شكل البيانات سواء فى المدخلات أو المخرجات ؟
* ماهو مستوى الدقة المطلوب للبيانات ؟
* ماهو عدد الكسور العشرية المطلوب استخدامه ؟
* هل من المتوقع تعطيل حركة البيانات فى أى وقت ؟

٧ - الموارد (Resources)

* ما هى المواد والأشخاص والموارد الأخرى المطلوبة لإنشاء واستخدام وصيانة النظام ؟
* ما هى المهارات المطلوب وجودها عند المصممين ؟
* ما هو حجم المكان أو الفراغ الذى سيتم شغله بواسطة النظام ؟
* ما هى متطلبات القدرة الكهربية والحرارة أو تكييف الهواء ؟
* هل هناك جدول زمنى (Timetable) للتصميم ؟
* هل هناك قيود على كمية النقود التى يتم إنفاقها على التطوير والمكونات والبرامج؟

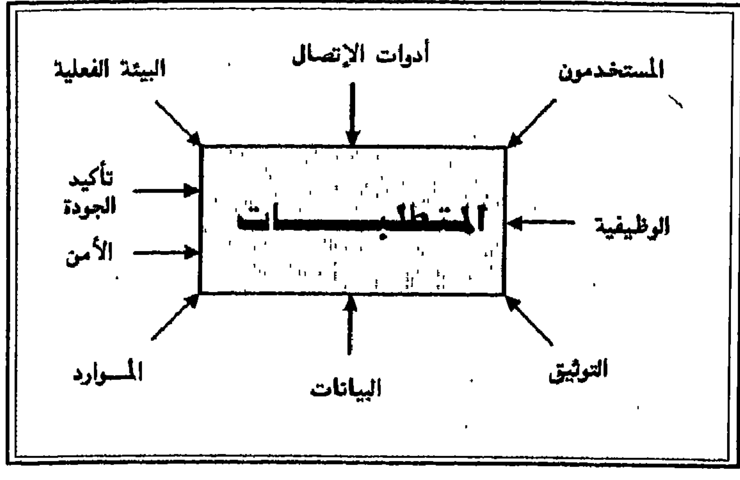
٨ - الأمن (Security)

* هل يجب السيطرة على الدخول إلى النظام أو إلى المعلومات ؟
* كيف سيتم عزل بيانات كل مستخدم عن المستخدمين الآخرين ؟
* كيف سيتم عزل برامج المستخدم عن باقى البرامج وكذلك عن نظام التشغيل ؟
* كيف سيتم عمل نسخ إحتياطية من برامج وبيانات النظام ؟
* هل يجب حفظ النسخ الإحتياطية فى مكان آخر ؟
* هل يجب اتخاذ إجراءات وقائية ضد الحريق والسرقة ؟

٩ - تأكيد الجودة (Quality Assurance)

* ما هى متطلبات الإعتمادية (Reliability) ؟
* هل يجب أن يقوم النظام بإكتشاف وعزل الأخطاء ؟
* ما هو الوقت المطلوب بين الأعطال ؟
* هل هناك حد أقصى للوقت المسموح لاستعادة النظام بعد تعطله ؟
* ما مدى استجابة النظام لأى تعديلات فى التصميم ؟
* ما مدى سهولة نقل النظام من مكان إلى آخر أو من نوع معين من الحاسبات إلى نوع آخر ؟

والشكل (٤١ - ٢) يوضح هذه الأنواع من المتطلبات



شكل (٤١ - ٢)

٤١ - ٣ خصائص المتطلبات

كما سبق أن أوضحنا فإن المتطلبات هى التى توضح مواصفات النظام من وجهة نظر العميل وهى التى يتم نقلها إلى مصممى النظام بالصورة المناسبة للتصميم. وفى الواقع فإن دور المتطلبات لايقف عند هذا الحد فهى تفيد أيضا فى قياس أداء النظام (Performance) وهذا يساعد فريق الإختبار على توضيح خصائص النظام ونقط القوة فيه باستخدام القياسات الكمية (Quantitative Measures). وهناك خصائص معينة يجب توافرها فى المتطلبات تتلخص فى الآتى :

١ - يجب أن تكون صحيحه (Correct) ولذلك يتم مراجعتها بواسطة العميل والمحلل للتأكد أنها موثقة بطريقة سليمة لا تحتمل الخطأ.

٢ - يجب أن تكون متطابقة (Consistent) وهذا يعنى عدم وجود تناقض بين المعلومات. فمثلا إذا كان أحد المتطلبات يحدد عدد المستخدمين الذين يستخدمون النظام فى وقت معين بما لايزيد عن عشرة وكان هناك أحد المتطلبات الأخرى الذى يقتضى استخدام عشرين مستخدم للنظام فى نفس الوقت يقال فى هذه الحالة أن المتطلبات غير متطابقة (Inconsistent).

٣ - يجب أن تكون كاملة (Complete) وهذا يعنى أن المتطلبات تغطى جميع الحالات ولا تترك مجالا للخطأ. فمثلا نظام المرتبات يجب أن يوضح ما يحدث عندما يحصل أحد الموظفين على أجازة بدون مرتب، أو عندما يأخذ أحد الموظفين علاوة .... الخ.

٤ - يجب أن تكون واقعية (Realistic) وهذا يعنى أن تكون قابلة للتنفيذ وليست خيالية.
٥ - يجب أن تكون مطلوبة (Needed) وهذا يعنى أن يكون العميل فعلا فى حاجة إلى هذه المتطلبات وأنها ضرورية. فمثلا قد يطلب العميل استخدام نوع محدد من الحاسبات بينما تكون هناك أنواع أخرى تحقق متطلبات العميل بكفاءة أكبر.
٦ - يجب أن تكون قابلة للتحقق أو التأكد (Verifiable) وهذا يعنى إمكانية اختبار النظام والتأكد من تحقيقه هذه المتطلبات.
٧ - يجب أن تكون سهلة التعقب أو الوصول (Traceable) وهذا يعنى أن كل وظيفة للنظام يمكن تعقبها للوصول إلى المتطلبات التى تحققها. فمثلا إذا أردنا مراجعة جميع المتطلبات الخاصة بالإتصالات (Communications) لايحتاج الأمر إلى قراءة كل المتطلبات الأخرى.

ولتوضيح أهمية هذه الخصائص فى اختبار المتطلبات وقياس أداء النظام نفرض أن أحد المتطلبات تم صياغته كالآتى :

" يجب أن يوفر النظام استجابة للإستفسارات فى الوقت الحقيقى "

فى هذه الحالة لا نستطيع تقدير قيمة الوقت الحقيقى ولكن إذا تم صياغة المطلوب بالطريقة الآتية :

" يجب أن يوفر النظام استجابة للإستفسارات فى مدة لاتزيد عن ثانيتين "

ففى هذه الحالة نعرف تماما كيف نختبر هذا الطلب.

وإذا فرضنا أن أحد النظم يطلب اتصال المستخدم بحاسب موجود على بعد عدة آلاف من الأميال ويطلب فى نفس الوقت أن يكون زمن الإستجابة (Response Time) هو نفس الزمن الخاص بالمستخدم الموجود فى نفس مكان الحاسب فى هذه الحالة يصبح هذا الطلب غير واقعى (Unrealistic).

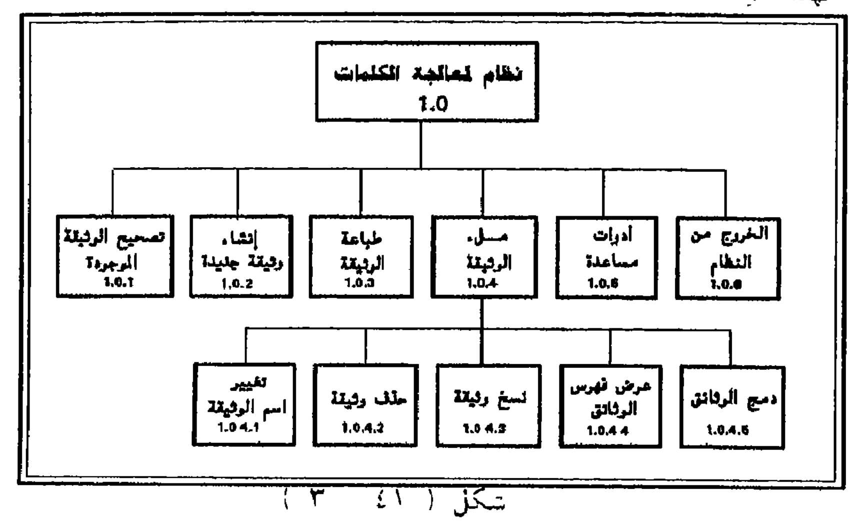
٤١ - ٤ وسائل وأدوات توصيف المتطلبات

هناك وسائل وأدوات متعددة لتوصيف المتطلبات تتراوح بين الوسائل اليدوية (Manual) والوسائل الآلية (Automated). وسبب تعدد هذه الوسائل هو محاولة الوصول إلى أحسن تنظيم قياسى (Standard) لطريقة توصيف المتطلبات وفى هذا الفصل

يتم توضيح أهم الوسائل والأدوات المستخدمة.

٤١ - ٤ - ١ خرائط هيبو (HIPO Charts)

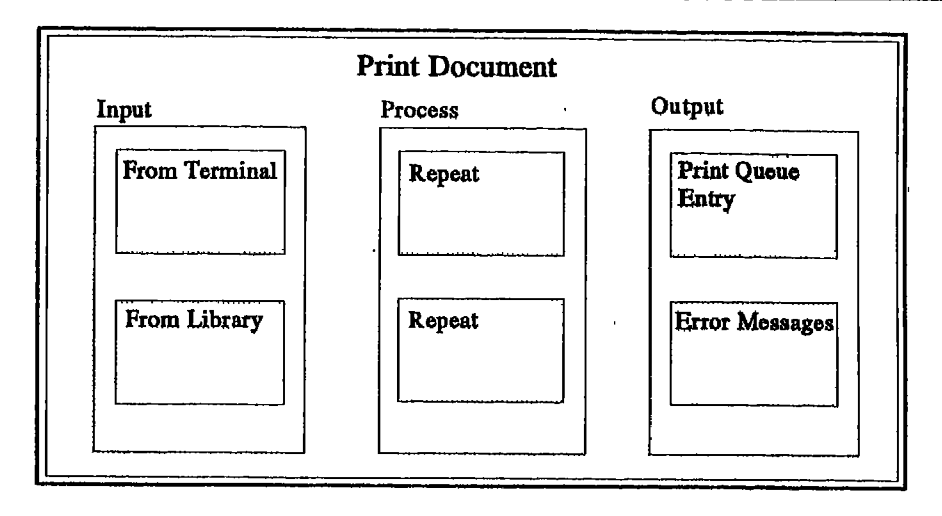
وهى وسيلة لتوصيف المتطلبات استخدمتها شركة(IBM) لمدة طويلة. وكلمة هيبو (HIPO) هى اختصار الكلمات التالية (Hierarchy and Input-Process-Output). وهذه الخرائط توضح علاقة وظائف النظام ببعضها حيث نبدأ أولا بتحديد وظائف النظام ثم يتم رسم شكل هرمى (Hierarchy) لهذه الوظائف بحيث تكون الوظائف الرئيسية فى أعلى الشكل الهرمى وتتفرع منها الوظائف الفرعية. والشكل (٤١ - ٣) يوضح المخطط الهرمى الخاص بنظام تنسيق كلمات (Word Processing). ويلاحظ من الشكل أن إسم الوظيفة يكون مكتوبا داخل المستطيل الخاص بها بالإضافة إلى الرقم الخاص بهذه الوظفة.



شكل (٤١ - ٣)

ويرتبط بهذا المخطط شكل آخر خاص بكل وظيفة فيه. هذا الشكل يوضح هذه الوظيفة والعملية المرتبطة بهذه الوظيفة (Process) ومخرجاتها (Outputs). أنظر شكل (٤١ - ٤).

ويلاحظ من الشكلين أن الشكل الأول يمثل المخطط الهرمى (H) أى (Hierarchy) والشكل الثانى يمثل المدخلات والعملية والمخرجات (IPO) أى (Input-Process-Output) ومن هذا جاءت تسمية المخطط. والشكل (٤١ - ٤) يوضح المخطط الخاص بالوظيفة رقم (1.0.3) فى مخطط الوظائف.



شكل (٤١ - ٤)

وهناك عدة عيوب لهذه الطريقة أهمها أنها توضح فقط المتطلبات الوظيفية (Functional) ولكنها تهمل المتطلبات غير الوظيفية (NonFunctional). والمقصود بالمتطلبات الوظيفية هى تلك المتطلبات التى تدخل فى صلب النظام. أما المتطلبات غير الوظيفية فهى المتطلبات التى تساعد على اختبار أداء النظام مثل القيود التى تفرض على قيم معينة. ومن عيوب هذه الطريقة أيضا أنها لاتتضمن وسائل الاختبار اللازمة وهذا يفرض على العميل أن يقوم بنفسه باختبار هذه المتطلبات بدقة والتأكد أنها تحقق مطالبه. كما أن العميل أيضا يكون مطالبا بعمل دراسات الجدوى (Feasibility Studies) على هذا النظام.

٤١ - ٤ - ٢ منهجية هندسة البرامج (SREM)

هذا المصطلح يعبر عن اختصار الكلمات الإنجليزية التالية :

(Software Requirement Engineering Methodolgy)

وهذه المنهجية تستخدم مع النظم الكبيرة المبنية على الوقت الحقيقى (Real Time) وهى نظم معقدة نتيجة القيود العديدة المفروضة على المتطلبات. وتعتمد هذه المنهجية على جزئين الأول خاص بتوصيف المتطلبات والثانى خاص بتحليل المتطلبات واستخراج التقارير. وتبدأ هذه المنهجية بكتابة المتطلبات بلغة تسمى لغة تدوين المتطلبات (Requirement Statement Language) وتختصر (RSL). ثم يتم تحليل هذه المتطلبات بواسطة نظام تحقيق المتطلبات الهندسى

(Requirements Engineering Validation System) ويختصر (REVS).

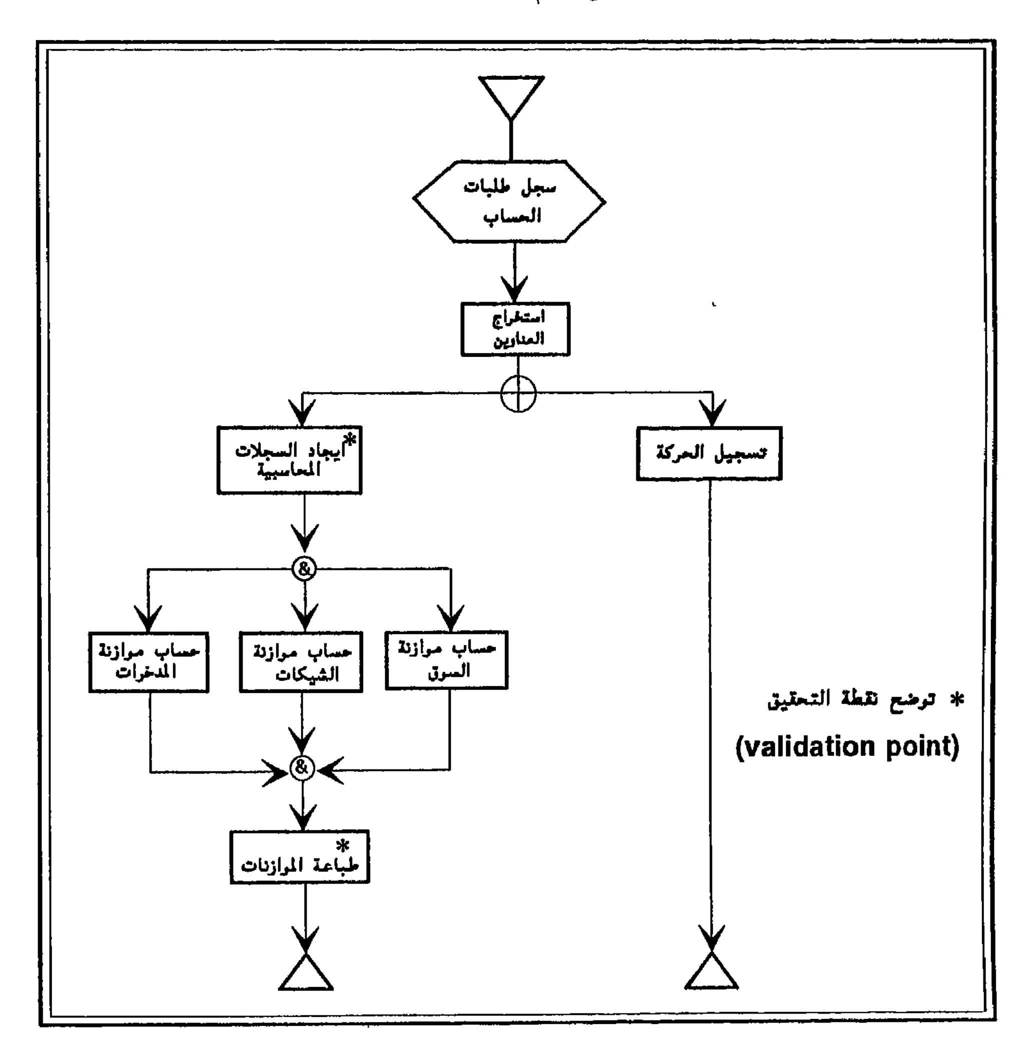
ويتم بواسطة لغة تدوين المتطلبات (RSL) توصيف تدفق البيانات (Data Flow) من خلال شبكة متطلبات (R-net) باستخدام الرسم والكتابة فى نفس الوقت. ويتم من خلال هذه الشبكة توصيف كيفية تحويل المدخلات إلى مخرجات. والشكل (٤١ - ٥) يوضح شكل شبكة المتطلبات فى نظام بنكى مباشر (On Line). ويلاحظ وجود دوائر تحتوى على علامة (+) وهى تعنى وجود شرط معين يتم بناء عليه التفرع إلى اليمين أو إلى اليسار. كما يلاحظ وجود دوائر تحتوى على علامة (&) وهى تعنى أن العمليات التالية لتلك الدوائر تتم على التوازى وبأى ترتيب فيما بينها. وبعد إنشاء مخططات الشبكة لكل المتطلبات، يتم ترجمة كل مخطط منها إلى الجمل الخاصة به بلغة (RSL).

ويلاحظ أن مخطط شبكة المتطلبات يوضح نقط المتطلبات لوظيفية (Functional) ولكن يمكن أيضا توضيح المتطلبات غير الوظيفية (Nonfunctional) مثل القيود المفروضة على بعض العمليات. فمثلا إذا نظرنا إلى الشبكة الموضحة فى الشكل السابق فقد يطلب العميل أن يتم طباعة كشف حساب العميل (Customer Account) فى خمس ثوان بعد الوصول إلى السجل الخاص به. ولتوضيح هذا القيد على المخطط يتم إضافة ما يسمى بنقطة التحقق (Validation Point). وهى نقطة يتم بواسطتها توضيح بداية ونهاية قياس المسار. فمثلا فى المثال السابق الخاص بالتطبيق البنكى يمكن وضع نقطة تحقق فى المستطيل الخاص بالبحث عن سجل الموازنة (Find Account Record) كما يمكن وضع نقطة تحقق أخرى فى المستطيل الخاص بطباعة الموازنة (Print Balance) ثم يتم تحديد فترة خمس ثوان لتمثل المسار من النقطة الأولى إلى النقطة الثانية.

وبعد استخدام لغة (RSL) فى ترجمة المتطلبات سواء كانت وظيفية أو غير وظيفية يتم استخدام نظام (REVS) فى تحليل هذه المتطلبات. ويقوم نظام (REVS) بترجمة الجمل الخاصة بلغة (RSL) لتكوين قاعدة بيانات تسمى نموذج التمثيل الدلالى للنظام (Abstract System Semantic Model) وتختصر (ASSM). وهذا النموذج يصحبه مجموعة من الأدوات التى تحلل محتوياته وتنتج مجموعة من التقارير التى توضح نتائج هذا التحليل.

وهناك تطوير لهذه المنهجية يؤدى إلى منهجية أخرى تسمى منهجية هندسة متطلبات النظم (Systems Requirement Engineering Methodology) وتختصر

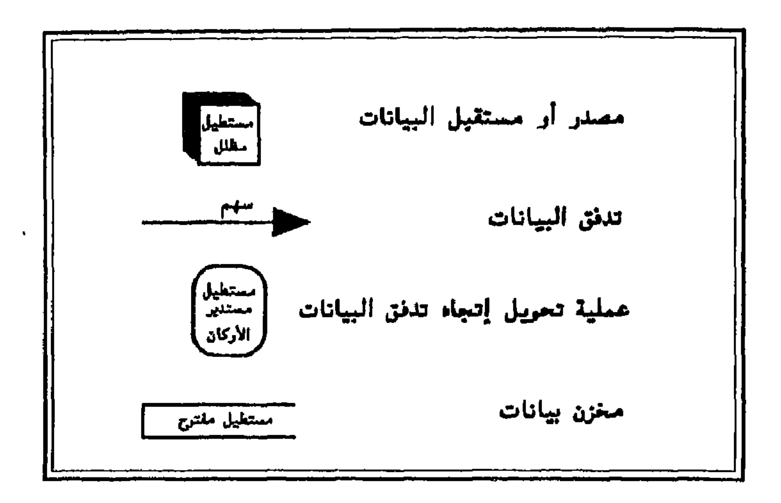
(SYSREM) وهى تضيف بعد الزمن إلى المنهجية السابقة. وتؤدى إضافة بعد الزمن إلى توضيح العلاقات الزمنية بين الأحداث وتساعد على توصيف أداء النظام واستجابته لأى مؤثرات خارجية. وبصفة عامة فإن هذه المنهجية والمنهجية السابقة يوفران عدة مزايا. الأولى أنه من السهل استخدام لغة (RSL) فى ترجمة المتطلبات إلى مجموعة تفصيلية من النشاطات والبيانات الموصفة جيدا. والثانية أن النظام يكون مقسما إلى أجزاء وظيفية منفصلة مما يسهل اختبارها. كما أن نظام (REVS) يستخدم بدائل مختلفة لمحاكاة المتطلبات وتوضيح جدوى النظام (Feasibility) وهذا يناسب النظم التى تكون مطمورة (Embeded) فى نظم أخرى كبيرة.



شكل (٤١ - ٥)

٤١ - ٤ - ٣ خرائط تدفق البيانات (Data Flow Diagrams)

خرائط تدفق البيانات تعد من أهم الوسائل أو الأدوات المنطقية المستخدمة فى توصيف متطلبات النظام. وهى تستخدم عادة أربعة رموز فى توضيح تدفق البيانات بين أجزاء النظام المختلفة كما يتضح من الشكل (٤١ - ٦). وهى مفيدة جدا لصمم النظام (System Designer) لأنه يستطيع بواسطتها إنشاء النموذج الفعلى (Physical Model) للنظام. وهى أيضا من الوسائل المتميزة لتوضيح وظائف النظام وعملياته للمستخدم وكذلك توضيح العلاقات بين مكونات النظام المختلفة.



شكل (٤١ - ٦)

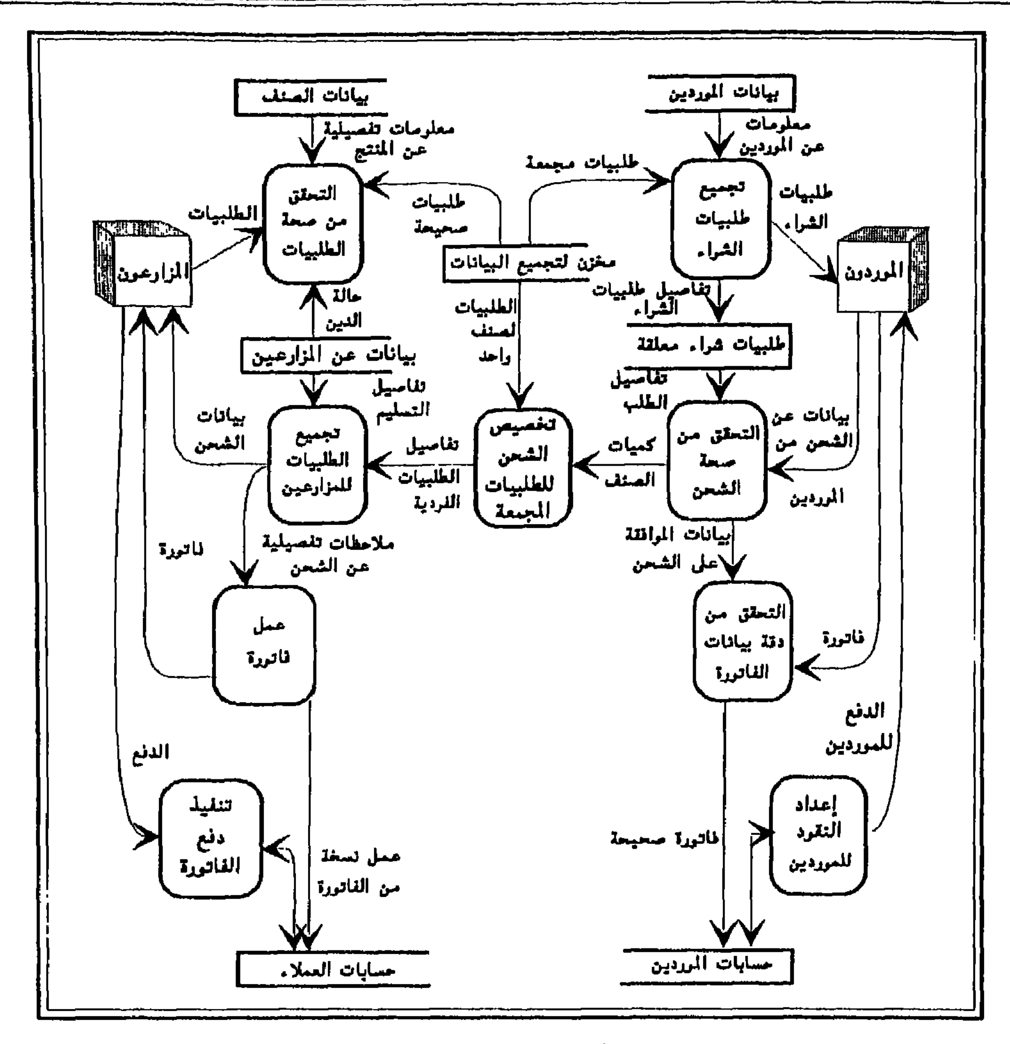
ولتوضيح خرائط التدفق يمكن دراسة مثال خاص بمنشأة لتوريد الأدوات والمواد الزراعية (Farming Supply Agency). هذه المنشأة تتيح للمزارعين طلب أسمدة أو حبوب أو آلات زراعية أو أشياء أخرى. ونظرا لضخامة حجم الطلبيات من المنشأة فإن سياستها تعتمد على عدم الإحتفاظ بأى مخزون من السلع. لذلك فإن المنشأة تقوم بتجميع الطلبيات مرة واحدة فى الأسبوع وعند وصولها من الموردين (Suppliers) تقوم بتوقيع طلبيات المزارعين. من ذلك يتضح أن النظام يحتوى على كيانين (Entities) الأول هو المزارعين (Farmers) والثانى هو الموردين (Suppliers). كما أن هناك مدخلات (Inputs) ومخرجات (Output) لكل كيان يوضحها الجدول التالى :

الكيان الخارجى	المدخلات القادمة منه	المخرجات المرسلة إليه
المزارعون	طلبيات تسديد	بيانات شحن فواتير
الموردون	بيانات شحن فواتير	طلبات شراء تسديد

والطلبيات (Orders) التى تصل من المزارعين يجب تحقيقها (Validate) للتأكد أنها خاصة بأصناف يمكن الحصول عليها من الموردين (Suppliers). ويجب أيضا اختبار كل طلبية (Order) للتأكد أن حساب المزارع (Credit) يغطى إجمالى هذه الطلبية. ولتحقيق ذلك فإنك تحتاج إلى مخزن بيانات يحتوى على بيانات عن كل السلع المتاحة وكذلك مخزن بيانات (Data Store) آخر يحتوى على بيانات المزارعين.

ومن خلال عملية تحقيق الطلبيات فإن بعضها سوف يتم إلغاؤها لأنها غير كاملة أو لأنها خاصة بأصناف غير متاحة أو لأنها خاصة بمزارعين ليس لديهم الحساب الكافى لتغطية هذه الطلبيات. بعد ذلك يتم تجميع الطلبيات فى مخزن بيانات الطلبيات المجمعة وذلك لأن هناك فارقا زمنيا بين استلام الطلبية وبين وصول الطلبيات إلى العدد الذى يسمح بإرسالها إلى المورد. وعند وصول الطلبيات إلى العدد المطلوب يتم تجميع أعداد كل صنف داخل هذه الطلبيات وعمل طلبات الشراء (Purchase Orders) الخاصة بها وإرسالها إلى المورد. وهذا يتطلب مخزن بيانات يحتوى على بيانات الموردين. وعندما يرسل المورد شحنة السلع المطلوبة فإنه يرسل مع هذه الشحنة بوليصة الشحن التى تحتوى على بيانات السلع المشحونة. وهذه البيانات يتم مقارنتها بطلبات الشراء للتأكد من إرسال الأصناف والكميات السليمة. وعندما تصبح جميع الأصناف متاحة يتم تجميع الطلبيات الخاصة بها وإرسالها إلى المزراعين. كما يتم إرسال بوليصة شحن (Shipping Note) مع كل طلبية توضح محتويات هذه الطلبية.

وأخيرا تقوم المنشأة باستلام الفواتير من كل مورد (Supplier) وتقوم بسداد هذه الفواتير. كما أن المنشاة تقوم أيضا بإرسال فواتير للمزارعين وإستلام المبالغ الخاصة بها ثم إدخال بيانات هذه المبالغ فى حسابات المزارعين. والشكل (٤١ - ٧) يوضح مخطط تدفق البيانات السابق شرحه.



شكل (٤١ - ٧)

ويجب ملاحظة أن مخطط تدفق البيانات بهذه الصورة يكون فى مرحلة المستوى العالى (High Level) ويكون مفيدا فى شرح وظائف النظام للمديرين وتوضيح أجزاء النظام التى يجب ميكنتها (استخدام الحاسب فى تنفيذها). ويلاحظ أيضا أنه لايتضمن أى تفاصيل فنية ولكنه يستخدم كأحد مدخلات عملية التصميم. حيث أن مصمم النظام يحصل منه على بيانات مايجب أن يحققه النظام (What the system should do) ليقرر كيف يحقق هذه المتطلبات (How the system does it).

والواقع أن مخطط تدفق البيانات يمر بعد ذلك بمراحل متعددة حيث يمكن تفجير (Explode) كل عملية لتوضيح مكوناتها وينتج عن هذا التفجير عادة مخطط تدفق بيانات جديد وهكذا حتى الوصول إلى أدق مكونات النظام. كما أن مخطط تدفق البيانات لايكون كافيا وحده وإنما تكون هناك أدوات أخرى لتوضيح محتويات مخازن البيانات (Data Stores) ولتوضيح التسلسل المنطقى للعمليات ..... الخ. وهذا ما سوف يتم إيضاحه فى الفصل الخاص بتصميم النظام.

٤١ - ٥ تحليل المتطلبات

أوضحنا فى الأجزاء السابقة أن أول مرحلة فى تطوير النظام هى تحديد متطلبات النظام ثم توصيف هذه المتطلبات باستخدام الوسائل التى سبق شرحها والتى من أهمها خرائط تدفق البيانات (DFD). كما أوضحنا أن خرائط تدفق البيانات تستخدم رموزا محددة للعمليات (Processes) ومخازن البيانات (Data Stores) وتدفق البيانات (Data Flow). وفى هذا الفصل يتم تحديد وسائل تحليل المتطلبات من خلال توصيف كل جزء من أجزاء خريطة تدفق البيانات ووسائل تبسيط هذه المحتويات.

٤١ - ٥ - ١ قاموس البيانات (Data Dictionary)

هل تستطيع التحدث باللغة الإنجليزية إذا لم تكن تعرف معانى الكلمات ودون الحاجة إلى قاموس (Dictionary) ؟ لاشك أن الإجابة ستكون بالنفى. فالقاموس هو الذى يوضح معانى الكلمات ويضع قواعد ثابتة وقياسية (Standard) للتحدث. وإذا لم يضع القاموس هذه القواعد فإن الكلمة الواحدة قد تصبح لها معان متعددة.

ونفس الشىء يمكن أن يقال بالنسبة لنظم المعلومات. فإذا لم يتوفر فى النظم الكبيرة قاموس بيانات (Data Dictionary) يصبح تطوير النظام عملية معقدة للغاية. حيث نجد المبرمجين والمصممين يبحثون عن محلل النظام لشرح معنى أحد عناصر البيانات المستخدمة فى النظام وهذا لايسبب إهدارا للوقت فقط ولكنه أيضا يسبب المتاعب لكل المشاركين فى تطوير النظام. حيث أن بعض عناصر البيانات يتغير إسمها عند انتقالها من المستخدم إلى المحلل إلى المصمم إلى المبرمج.

وقبل أن نبدأ فى شرح مكونات قاموس البيانات سنقوم بتوضيح أهم المصطلحات المرتبطة بالبيانات وبخريطة تدفق البيانات وهى كالآتى:

١ - عناصر البيانات (Data Elements)

وهى تمثل أصغر وحدة بيانات لها معنى مرتبط بالهدف من النظام فالتاريخ (Date) هو عنصر بيانات بالنسبة لمعظم المحللين. ولكن بعض المحللين قد يعتبرونه هيكل بيانات (Data Structure) مكون من اليوم والشهر والسنة كعناصر بيانات. وذلك عندما تكون هناك حاجة للتعامل مع جزء من التاريخ فقط.

٢ - هياكل البيانات (Data Structures)

وهى تتكون من مجموعة من عناصر البيانات أو هياكل البيانات أومن خليط من الإثنين.

٣ - تدفق البيانات (Data Flow)

تدفق البيانات يقصد به المسار الذى يوضح حركة البيانات من خلال خريطة التدفق.

٤ - مخازن البيانات (Data Stores)

وهى الأماكن التى يتم فيها تجميع هياكل البيانات إلى حين الحاجة إليها. ويمكن القول أن تدفق البيانات (Data Flow) يعبر عن هياكل البيانات فى حالة الحركة بينما مخازن البيانات تعبر عن هياكل البيانات فى حالة السكون.

والشكل (٤١ - ٨) يوضح تدفق هيكل البيانات الذى يسمى (Order) والذى يحتوى على هياكل بيانات مثل(Order - ID) ، (Customer - Details) ، (Phone) ، ... إلخ. كما أن هياكل البيانات تحتوى على عناصر بيانات مثل (Order - Date) ، (First - Name) ، (Last Name) ، ... إلخ.

والشكل (٤١ - ٩) يوضح الشكل الهرمى لتوصيف البيانات فى المستويات المختلفة.

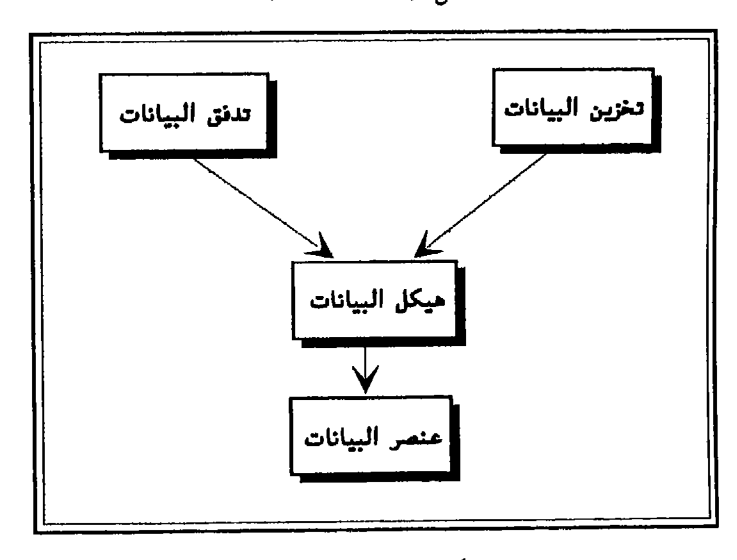
٤١ - ٥ - ٢ مدير البيانات (Data Adminstrator)

كما أوضحنا فإن قاموس البيانات يمثل المخزن المركزى للبيانات بالنسبة للمحللين والمصممين والمبرمجين الذين يعملون فى تطوير نظام معين. وحيث أن هذا

القاموس يكون مركزيا فإن السيطرة عليه تتم بواسطة شخص أو مجموعة من الأشخاص الذين يطلق عليهم إسم مديرى البيانات (Data Administrators).

```
ORDER
   ORDER-ID
      ORDER-DATE
      CUSTOMER-ORDER-NUM
   CUSTOMER-DETAILS
      ORGANIZATION-NAME
      PERSON-AUTHORIZING
        FIRST-NAME
        LAST NAME
      PHONE
        AREA-CODE
        EXCHANGE
        NUMBER
        EXTENSION
     SHIP-TO-ADDRESS
        STREET
        CITY-COUNTY
        STATE-ZIP
      BILL-TO-ADDRESS
        STREET
        CITY-COUNTY
        STATE-ZIP
      BOOK-DETAILS
        AUTHOR-NAME
        TITLE
        ISBN
        LOCN
        PUBLISHER-NAME
```

شكل (٤١ - ٨)



شكل (٤١ - ٩)

ومدير البيانات يكون مسئولا عن السيطرة على المدخلات والتعديلات التى يتم إجراؤها على القاموس. وهو إلى جانب ذلك يجب أن يسهل على أى شخص يعمل فى تطوير النظام الوصول إلى أى بيان فى القاموس وأن يكون هذا البيان محدثا (Up-to-date) أى أنه بصفة عامة يجب أن يمنع أى تناقض أو اختلاف ينتج عن عدم توحيد تعريف البيانات بالنسبة للمحللين.

٤١ - ٥ - ٣ محتويات قاموس البيانات

يحتوى قاموس البيانات على توصيف دقيق لأجزاء النظام التى تشمل سبعة مكونات رئيسية وهى عناصر البيانات (Data Elements) وهياكل البيانات (Data Structures) وتدفقات البيانات (Data Flows) ومخازن البيانات (Data Stores) والعمليات (Processes) والكيانات الخارجية (External Entities) وذلك بالإضافة إلى قاموس مصطلحات المستخدم. ويمكن إنشاء هذا القاموس بإحدى طريقتين ، الأولى يدويا والثانية آليا أى باستخدام الحاسب والطريقة الثانية هى الأفضل. أنظر الشكل (٤١ - ١٠).

Entry	Type
SALARY Annual salary for both hourly and salaried employees	Data element
SALARY-CHANGE Input from management to alter salary	Data flow
SHOW STATUS Display current status for a given employee	Process
SOC-SEC-NUM Social security number	Data element
SS # Alias for social security number	Data element
STATE-ZIP Combination of state, province, country, code	Data structure
STATUS Details of all non-salary current information about employee	Data structure
STATUS - CHANGE All non-salary changes, e.g. address, dependents, job, training	Data flow
STATUS - HISTORY Holds data on changes of employees status and salary for entire career with company	Data store

شكل (٤١ - ١٠)

٤١ - ٥ - ٤ خصائص قاموس البيانات

نظرا لأهمية قاموس البيانات بالنسبة للنظم الكبيرة بصفة خاصة فإن هناك خصائص معينة يجب توافرها فيه. وسوف يتم توضيح أهم هذه الخصائص فى هذا الجزء.

أ - التقارير التحليلية (Analysis Reports)

فى مرحلة التحليل التفصيلى (Detailed Analysis) لأى عملية فى خريطة التدفق أو لأى هيكل بيانات فإننا نحتاج إلى معرفة التفاصيل الدقيقة عن هذه العملية أو هذا الهيكل. وهذا يعنى أن القاموس يكون مخزنا فى قاعدة بيانات مع استخدام أحد برامج إدارة قواعد البيانات فى تشغيلها.

ب - المراجعة والتعديل (Review & Chanage)

بعد الإنتهاء من تحديد تدفقات البيانات وإدخال معظم بيانات القاموس يجب مراجعة القاموس عدة مرات. وخلال عمليات المراجعة فإننا نجد أن بعض عناصر البيانات لم يتم إدخالها أو أن بعض العناصر الموجودة تحتاج إلى تعديل. لذلك فإننا نحتاج إلى تعديل جزء من قاموس البيانات. ولكن المشكلة التى تواجهنا فى هذه الحالة أن تعديل بعض عناصر البيانات قد يؤثر فى بيانات أخرى فى القاموس. لذلك فإننا نحتاج إلى تعديل كل العمليات وهياكل البيانات التى تتأثر بهذا العنصر. وهذا يعنى أن القاموس يجب أن يتيح إمكانية البحث عن كل البيانات التى تتأثر بأى تعديل فى عناصر البيانات وإجراء هذا التعديل عليها.

جـ - القدرة على إيجاد الإسم من التوصيف

إذا فرضنا أننا نعلم أن هناك متوسط مشتريات كل ثلاث شهور تم إدخاله لكل عميل. وأننا لانعلم على وجه اليقين إذا كنا أدخلنا هذا البيان فى قاموس البيانات أم لا ، كما أننا لانعرف إسم هذا البيان فكيف نسترجع هذا البيان؟ فإما أننا سوف نحتاج إلى مدير بيانات يعرف إسم كل عنصر بيانات موجود فى القاموس - وهذا صعب جدا، وإما ان القاموس نفسه يتيح لنا البحث عن أى بيان عن طريق بيانات توصيف هذا البيان. وفى الحالة الثانية فإن ذلك يتم عن طريق تحديد كلمات حاكمة (Keywords) مع كل عنصر بيانات يمكن بواسطتها استرجاع هذا العنصر. فمثلا يمكن عرض كل عناصر البيانات التى تحتوى على الكلمات الحاكمة (Cust) ،

(Purch) ، (Av).

د - الكمال والتكامل

بعد الإنتهاء من إنشاء قاموس البيانات فإن هناك مجموعة من الأسئلة يجب أن تسألها لنفسك وهى :

١- هل هناك تدفقات بيانات تم توصيفها دون أن يكون لها مصدر (Source) أو جهة وصول (Destination) ؟
٢- هل هناك عناصر بيانات موجودة فى مخازن البيانات (Data Stores) ولاتظهر فى مدخلات مخازن البيانات ؟
٣- هل هناك عناصر بيانات موجودة داخل عملية (Process) ولاتظهر فى مدخلات هذه العملية ؟
٤- هل هناك عناصر بيانات تظهر فى مدخلات عملية معينة بينما لاتظهر داخل التسلسل المنطقى للعملية أو فى مخرجات هذه العملية ؟

٤١ - ٥ - ٥ مخزن البيانات (Data Store)

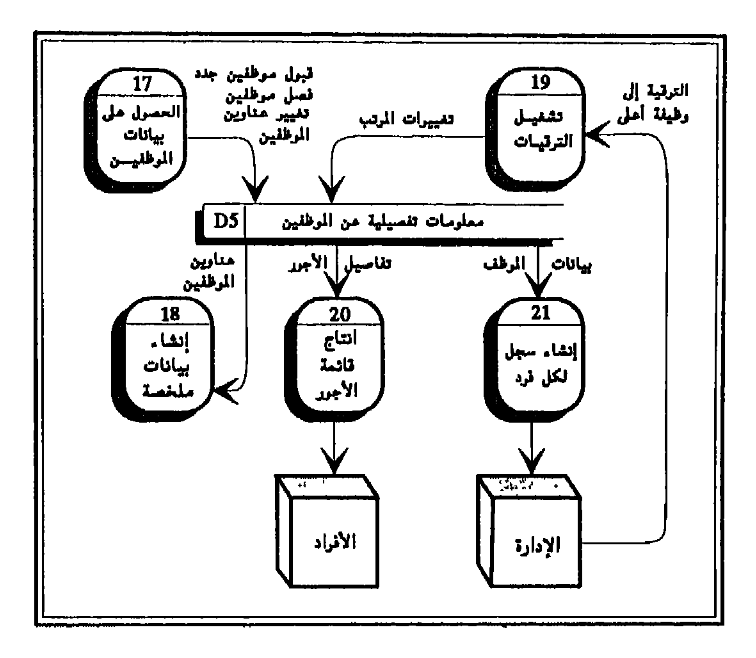
كما سبق أن أوضحنا فإن مخزن البيانات يحتوى على هياكل البيانات الساكنة. وتحديد محتويات مخزن البيانات ليس سهلا وإنما يتطلب معرفة جميع مدخلات ومخرجات هذا المخزن. كما يتطلب تطبيع البيانات المخزنة (Normalization) حتى تصبح خالية من أى تكرار أو عيوب أخرى كما سبق الإيضاح.

٤١ - ٥ - ٥ - ١ تحليل المدخلات والمخرجات

عادة يمكن استنتاج محتويات مخزن البيانات من مدخلات ومخرجات هذا المخزن. فإذا علمت مخرجات هذا المخزن فإنك ستعرف أقل محتويات يجب أن تكون موجودة به وذلك لأن أى مخرجات من المخزن يجب أن تكون موجودة أصلا به. والخطوة الثانية هى اختبار مدخلات هذا المخزن للتأكد من أن جميع البيانات التى دخلت إلية خرجت منه لاستخدامها فى عمليات أو هياكل بيانات أخرى. وهذا يتيح لك التأكد أن كل بيان موجود فى مخزن البيانات سوف يستخدم مرة ثانية من خلال خريطة تدفق البيانات. ولتوضيح ذلك ندرس المثال التالى

مثـــال

الشكل (٤١ - ١١) يوضح جزءا من خريطة تدفق بيانات (DFD) خاصة بنظـام شئـون الأفـراد (Personnel System). فـإذا نظـرنـا إلى المخـزن (D5 - Employee - Details) فإننا نستطيع مـن تدفقات البيانات والعمليات المرتبطة بها أن نستنتج أنه يجب أن يحتوى على بيانات الإسم (Name) والعنوان (Address) والمرتب (Salary) لجميع الموظفين (Employees). ولكن هل هذا يكفى لتحديد محتويات المخزن (D5) ؟



شكل (٤١ - ١١)

للإجابة على هذا السؤال يجب أولا أن ندرس مدخلات هذا المخزن ومخرجاته. حيث نلاحظ أن هناك خمسة هياكل بيانات تتدفق من وإلى المخزن (D5). فإذا نظرا إلى محتويات هذه الهياكل فإننا نستطيع مقارنة عناصر البيانات الداخلة إلى هذا المخزن وعناصر البيانات الخارجة منه. والشكل (٤١ - ١٢) يوضح هذه المقارنة.

والخطوة التالية هى عمل مسودة (Draft) لمحتويات المخزن (D5) بناء على التحليل السابق. والمحاولة الأولى سوف تشبه الموضح بالشكل (٤١ - ١٣)

هياكل البيانات الموجودة فى :

تدفق البيانات من المخزن (D5)

(D5 - 18) عنوان موظف
NAME
ADDRESS

(D5 - 20) معلومات تفصيلية عن موظف
NAME
PERSONNEL-NO
CURRENT-SALARY

(D5 - 21) سجل الموظف
NAME
PERSONNEL-NO
DATE-HIRED
JOB-HISTORY*
  JOB-TITLE
  DATE-EFFECTIVE
SALARY-HISTORY*
  SALARY
  DATE-EFFECTIVE
  [REVIEW-SUMMARY]
(لكل الموظفين فى المؤسسة)

تدفق البيانات الى المخزن (D5)

(17 - D5) موظف جديد
DATE-HIRED
NAME
PERSONNEL-NO
ADDRESS
JOB-TITLE
SALARY-START

(17 - D5) فصل موظف
NAME
PERSONNEL-NO

(17 - D5) تغيير عنوان الموظف
NAME
PERSONNEL-NO
ADDRESS-OLD
ADDRESS-NEW

(19 - D5) تغيير المرتب
NAME
PERSONNEL-NO
SALARY-OLD
SALARY-NEW
DATE-EFFECTIVE

شكل (٤١ - ١٢)

STRUCTURE OF D5

Data structure /element

NAME
PERSONNEL-NO
ADDRESS
CURRENT-SALARY
DATE-HIRED
JOB-HISTORY*
  JOB-TITLE
  DATE-EFFECTIVE
SALARY-HISTORY*
  SALARY
  DATE-EFFECTIVE
  [REVIEW-SUMMARY]

شكل (٤١ - ١٣)

وعند مقارنة مدخلات مخزن البيانات (D5) فإننا نلاحظ الآتى :

١ - هيكل البيانات (Employee - Address) الذى يخرج من مخزن البيانات يتطلب وجود العنوان الحالى للموظف. وهيكل البيانات

(Address - Change) الذى يدخل إلى مخزن البيانات يعطى العنوان القديم (Address - Old) والعنوان الجديد (Address - New) ولكن هل نحتاج حقيقة إلى العنوان القديم ؟ واضح من مخرجات مخزن البيانات أننا لانحتاج إليه.

٢ - هيكل البيانات (Employee - History) الذى يخرج من مخزن البيانات يتطلب بيانات المرتب المتغيرة وليس المرتب السابق (Old - Salary) والمرتب الجديد (New - Salary) الموجودين فى هيكل البيانات (Salary - Chanage) الذى يدخل إلى مخزن البيانات.

٣ - هيكل البيانات (Job - History) يتطلب بيانات التوظيف المتغيرة. ولكن يلاحظ أن عنصر البيانات (Job - Title) يدخل مرة واحدة لكل موظف ولايتغير من خلال خريطة التدفق. وهذا أحد أخطاء المخطط التى يجب علاجها.

والشكل (٤١ - ١٤) يوضح محتويات مخزن البيانات بعد إدخال بيانات الموظفين.

NAME	PERSONNEL NUMBER	ADDRESS	CURRENT SALARY	DATE HIRED	JOB HISTORY		SALARY HISTORY		REV. SUM
					JOB-TITLE	DATE-EFF	SALARY	DATE-EFF	
Tarek Fahmy	26622	40-hegaz Hiliopolis	1800	12/1/84	Supervisor	9/1/86	1800	12/1/86	4
					Sen Prog	4/15/85	1500	12/1/85	3.5
					Programmer	12/1/84	1450	4/15/85	
Hasan Foad	30661	34-shobra street	2500	6/15/83	Manager	11/1/85	2500	6/15/86	4 5
					Shift-loader	12/15/84	2000	11/1/85	4.8
					Operator	7/1/84	1500	12/15/84	
					Typest	6/15/83	1200	6/15/84	4 5
Saleh Hasan	20327	21-sleem-street	850	1/1/85	Mail-clerk	1/1/85	850	1/1/86	2.8
							820	1/1/85	

شكل (٤١ - ١٤)

ويلاحظ من الشكل أن كل تغير فى وظيفة الموظف (Job - Title) يصحبه تغير فى مرتبه (Salary - History).

٤١ - ٥ - ٥ - ٢ تبسيط محتويات مخزن البيانات

بعد تحديد محتويات المخزن المبدئية يجب تبسيط هذه المحتويات عن طريق إجراء بعض العمليات المنطقية عليه. فمثلا يلاحظ فى المخزن (D5) أن هناك بعض البيانات المكررة حيث نجد أن عنصر البيانات (Current - Salary) يعطى دائما أحدث قيمة لعنصر البيانات (Salary - History) وبالتالى ليس هناك حاجة لتخزين كل منهما معا. وكذلك يلاحظ أن عنصر البيانات (Date - Hired) يعطى دائما أقدم تاريخ فى هيكل البيانات (Job - Title) وهيكل البيانات (Salary - History). وهذا يعنى أن هذا البيان مكرر فى ثلاثة مواضع. لذلك فالأفضل ضم هيكل البيانات (Job - Title) وهيكل البيانات(Salary - History) لتكوين هيكل بيانات جديد نسميه مثلا (Salary - Title - History) ويحتوى على عنصر البيانات (Date - Of - Change) الذى يحتوى على تاريخ أى تغيير يحدث سواء فى الوظيفة (Job) أو المرتب (Salary) أى أن هيكل البيانات الجديد يصبح كالآتى :

Salary - Title - History
Date - Of - Change
Job - Title
Salary
[REVIEW - SUMMARY]

فلوطبقنا هذا الهيكل على (طارق فهمى) تظهر البيانات كالآتى:

Date-Of-Change	Job - Title	Salary	Review-Summary
08/01/86	Supervisor	1800	4
12/01/85	Sen.Prog.	1500	3.5
12/01/84	Programmer	1450	

ولكن يجب ملاحظة أن تبسيط هيكل البيانات بالصورة السابقة يكون على حساب تعقيد أكبر فى التسلسل المنطقى لبعض العمليات التالية.

فمثلا فى العملية رقم (20) فى الشكل (٤١ - ١١) وهى العملية (Produce - Salary - listing) لايكفى إسترجاع بيانات الإسم (Name) والرقم الشخصى (Personal - Number) والمرتب الحالى (Current - Salary) مباشرة من مخزن البيانات وإنما يتطلب الأمر البحث عن أحدث بيان تم إدخاله فى هيكل البيانات (Salary - Title - History) وإحضار المرتب الحالى(Current Salary) . ولكن هذا التعقيد فى التسلسل المنطقى للعملية يقابله ثلاثة مميزات تتلخص فى الآتى :

١- الحصول على مخزن بيانات بسيط وصغير

٢- عند إدخال أى تغيير فى المرتب فإن المرتب الجديد يتم إدخاله فى مكان واحد (Salary) وليس فى مكانين كالحالة السابقة (Current - Salary) و (Old - Salary). وهذا يقلل إحتمالات الخطأ.

٣ - التأمين ضد أى تغيير. فمثلا لو نظرنا إلى العملية رقم (21) (Produce - Individual - Profile). فإننا نجد أن هيكل البيانات المبدئى الموضح بالشكل (٤١ - ١٣) يعنى أن المستخدم يريد تقارير منفصلة عن الوظائف (Job - Title) وتاريخ المرتب (Salary - History) وهذا يوضح سبب تصميم هيكل البيانات المبدئى بهذا الشكل. ولكن ماذا يحدث لو غير المستخدم رأيه وأراد عرض قائمة مشتركة للوظائف وتاريخ المرتب مرتبة حسب التواريخ ؟ فإن الهيكل المبدئى يتطلب عمل دمج بين الهيكلين (Job - Title) و (Salary - History) بالإضافة إلى إجراء عملية الترتيب (Sorting) وذلك من خلال التسلسل المنطقى للبرنامج. ولكن باستخدام الهيكل المعدل يصبح إجراء العملية المنطقية أسهل. وليست عملية تبسيط محتويات مخزن البيانات هى العملية الوحيدة التى يمكن إجراؤها للحصول على أبسط صورة لهيكل البيانات وإنما هناك عملية أخرى تسمى تطبيع البيانات (Normalization) سيتم شرحها فى الجزء التالى

٤١ - ٥ - ٥ - ٣ تطبيع البيانات (Normalization)

معظم قواعد البيانات أو الملفات بصفة عامة تتعرض للتغيير بصفة مستمرة حيث يتم إضافة عناصر بيانات أو هياكل بيانات جديدة كما تتغير العلاقات (Relations) بين عناصر البيانات. فمثلا قد تضطر إلى تقسيم سجل معين إلى

جزئين أو قد تحتاج إلى تغيير حقل المفتاح (Key field) للوصول إلى عنصر بيانات معين. وهذا قد يضطرنا ليس فقط إلى تغيير الملفات ولكن تغيير البرامج أيضا. ولتقليل هذه المشاكل يجب العناية الفائقة بالتصميم المبدئى لهياكل البيانات. لذلك يتم تنفيذ مجموعة من مراحل تطبيع البيانات (Normalization). بهدف التخلص من البيانات المتكررة وكذلك التخلص من الحقول المعتمدة جزئيا على حقل المفتاح (Key Field).

٤١ - ٥ - ٦ العمليات (Processes)

من الوسائل التقليدية لتوصيف العمليات إستخدام اللغة الإنجليزية فى تلخيص التسلسل المنطقى لكل عملية. ولكن هناك عيوب كثيرة لهذا الوسيلة منها أن اللغة الإنجليزية لاتستطيع توصيف التسلسل المنطقى للعملية بوضوح لأن الكلمات ليس لها معان ثابتة أو منفردة لذلك يتم فى هذا الجزء شرح بعض الوسائل المستخدمة فى توصيف العمليات.

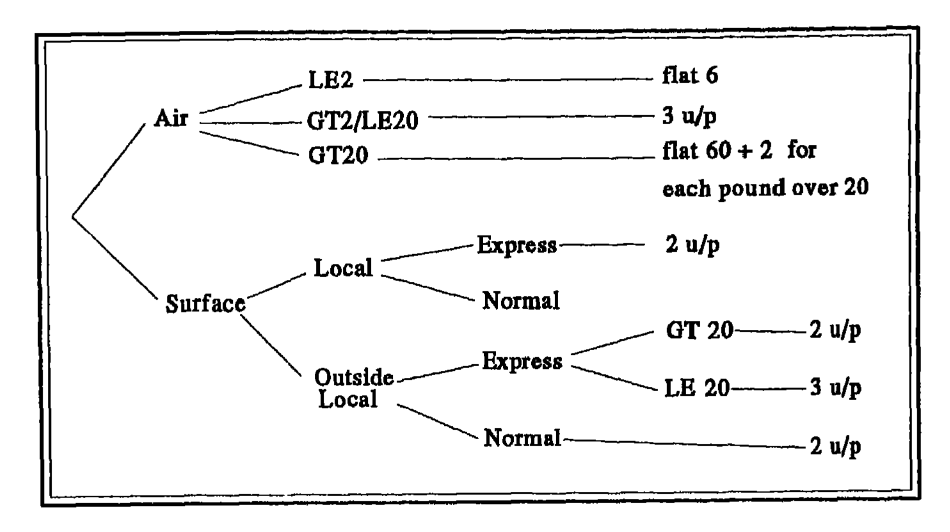
٤١ - ٥ - ٦ - ١ شجرة القرارات (Decision Tree)

نفرض أن شركة معينة تقوم بشحن طرود من الكتب إلى العملاء، وتتقاضى منهم تكاليف الشحن. وتكاليف الشحن يتم حسابها بالوحدة (Unit) التى تساوى (50 Cents). والسياسة العامة الخاصة بالشركة يمكن تلخيصها فى الآتى :

الشحن الجوى تعتمد تكاليفه على وزن الشحنة. والمعدل الأساسى هو (٣ وحدات لكل رطل) تقل إلى (٢ وحدة لكل رطل) للوزن الزائد عن(٢٠ كجم)· بحد أدنى ٦ وحدات. والشحن السطحى (Surface Freight) يتكلف (٢ وحدة لكل رطل) وذلك للتسليم السريع. وهذا المعدل يطبق فقط إذا كان تسليم الشحنة يتم داخل المنطقة المحلية (Local Area). أما إذا كان تسليم الشحنة يتم خارج المنطقة المحلية والشحنة تزن أكثر من ٢٠ رطل فإن تكاليف الشحن السطحى (Surface Freight) تكون مساوية لتكاليف التسليم السريع. والتسليم المعتاد للشحنة الأقل من ٢٠ رطل هو (٢ وحدة لكل رطل) مع وحدة لكل رطل فى حالة الشحن السطحى السريع.

هذه السياسة تمثل الإطار العام للعملية الخاصة بحساب تكاليف الشحن. وهى رغم أنها قد تبدو معقدة بعض الشىء إلا أن تمثيلها بواسطة شجرة القرارات (Decision Tree) يقلل من درجة تعقيدها إلى حد كبير. والشكل (٤١ - ١٥)

يوضح شجرة القرار الخاصة بهذه العملية.



شكل (٤١ - ١٥)

ويلاحظ من الشكل أن هناك ثلاثة رموز مستخدمة فيه وهى الآتى :

الرمز (GT) وهو إختصار (Greater Than)
الرمز (LE) وهو إختصار (Less than or Equal to)
الرمز (U/P) وهو إختصار (Units Per Pound)

٤١ - ٥ - ٦ - ٢ الشفرة الزائفة (Pseudocode)

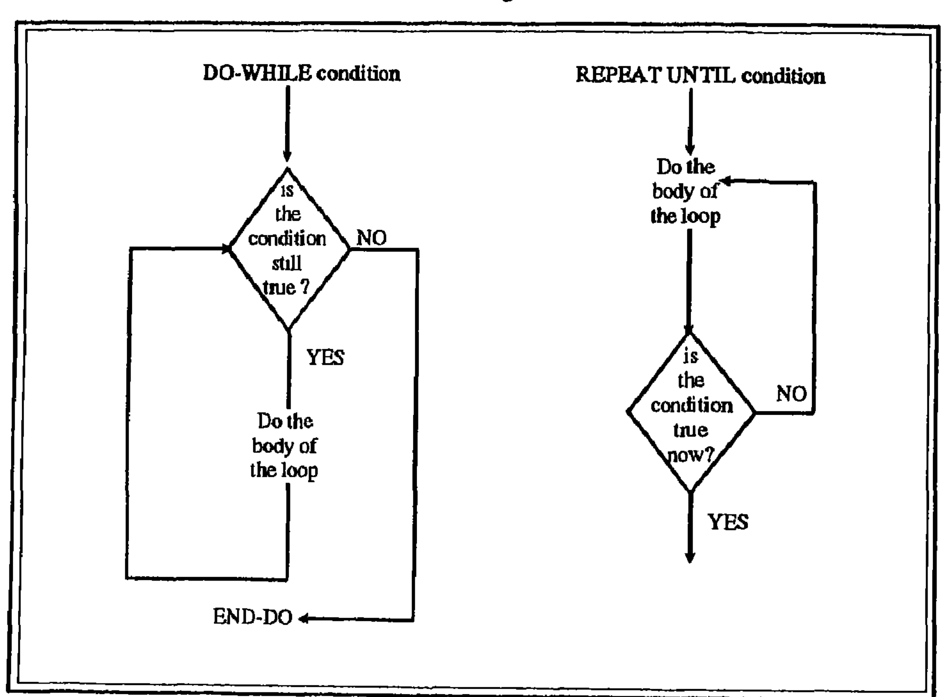
الشفرة الزائفة (Pseudocode) هى مرحلة متوسطة بين كود البرنامج المكتوب بأى لغة من لغات البرمجة وبين اللغة الإنجليزية. أى أنها توصيف للتسلسل المنطقى للبرنامج دون الدخول فى تفاصيل أوامر اللغة أو القواعد (Syntax) الخاصة بها. فمثلا إذا أردنا كتابة برنامج يسمى (Generate Invoice) يقوم بقراءة ملف يسمى (Todays - Shipment) فإن الشفرة الزائفة لهذا البرنامج تكون كالموضح بالشكل (٤١ - ١٦).

ويجب ملاحظة أن الحلقة التكرارية (Do - While) تختلف عن الحلقة التكرارية (Repeat - Until) فى البرنامج المكتوب بالشفرة الزائفة. والشكل (٤١ - ١٧) يوضح الفرق بين الإثنين حيث نلاحظ أن اختبار الشرط فى حالة

الحلقة (Do - While) يتم قبل تنفيذ الحلقة. بينما يحدث العكس مع الحلقة .(Repeat - Until)

```
Initialize the program (open files, set counters)
Read the first order - record
DO - WHILE there are more order records
        DO - WHILE there  are more items on th order
                Compute ITEM - TOTAL
                Add ITEM - TOTAL to INVOICE-TOTAL
        END - DO
        Compute discount
        Compute shipping and handling fee
        Compute invoice - net, total - payable
        Print invoice
        Write invoice to accounts - receivable file
        Add invoice - detail to summary counters
        Read next order record
END - DO
Print summary of day's invoices
Terminate program
```

شكل (٤١ - ١٦)



شكل (٤١ - ١٧)

والشفرة الزائفة هى وسيلة فعالة فى توصيف التصميم الفعلى (Physical Design) حيث يسهل على المبرمج ترجمتها إلى لغة (كوبول) أو أى لغة من اللغات عالية المستوى.

٤١ - ٥ - ٦ - ٣ جداول القرارات (Decision Tables)

كما سبق أن أوضحنا فى شجرة القرارات (Decision Tree) فإن توصيف التسلسل المنطقى للعمليات بهذه الطريقة يعتبر مفيدا جدا للمستخدم والمبرمج. ولكن هناك مشكلة واحدة فى استخدام هذه الوسيلة وهى أنها تفتقد المرونه فى حالة دمج عدد من الشروط. وهذا ما يتوفر فى جداول القرارات (Decision Tables) فإذا فرضنا العملية الممثلة بالجملة التالية :

" العملاء الذين يحققون حجم مشتريات يزيد عن (10.000 $) فى العام ولهم تاريخ تسديد جيد (Good Payment History) أو الذين يتعاملون معنا منذ أكثر من (٢٠) عاما يجب أن تكون لهم أولوية فى المعاملة "

فإن الشكل (٤١ - ١٨) يوضح جدول القرارات الممثل لهذه العملية.

c 1: more than $ 10.000 a yr?	Y	Y	Y	Y	N	N	N	N
c 2 : good payment history?	Y	Y	N	N	Y	Y	N	N
c 3 : With us more than 20 yrs?	Y	N	Y	N	Y	N	Y	N
a 1 : Priority treatment	X	X	X					
a 2 : normal treatment				X	X	X	X	X

شكل (٤١ - ١٨)

وفى هذا النوع من جداول القرارات يتم كتابة الأحداث (Actions) التى يتم تنفيذها كنتيجة للقرارات فى القسم السفلى الأيسر من الجدول. وهذا القسم يسمى قسم الأحداث (Action Stub) ويتم كتابة الشروط كأسئلة تحتمل

الإجابة بنعم (Y) أو لا (N) فقط وهذا يتيح الدمج بين عدة شروط عن طريق تمثيلها بقائمة من (Y) و (N). والعمود الواحد من الجدول الذى يمثل مجموعة من الشروط والأحداث يسمى قاعدة (Rule). فمثلا الشكل (٤١ - ١٩) يوضح القاعدة رقم (٣) (Rule 3).

وهذه القاعدة تماثل الجملة التالية

"If the customer does more than $10,000 business
and
he does not have a good payment history
and
he has been with us more than 20 years"

وعلامة X فى الجزء السفلى من الجدول تمثل جملة جواب الشرط لجملة (IF) السابقة كالآتى :

"Then he gets Priority Treatment"

c 1: more than $ 10.000 a yr?			Y					
c 2 : good payment history?			N					
c 3 : With us more than 20 yrs?			Y					
a 1 : Priority treatment			X					
a 2 : normal treatment								

شكل (٤١ - ١٩)



الفصل الثانى والأربعون

تصميم النظام



قبل أن نخوض فى شرح تصميم النظام والوسائل والأدوات المستخدمة فيه سوف نوضح الفرق بين تحليل النظام (System Analysis) وتصميم النظام (System Design) ودور كل من المحلل والمصمم. وقد سبق أن عرفنا تحليل النظام بأنه عمل نموذج منطقى (Logical Model) للنظام بالإضافة إلى تحديد مجموعة من الأهداف التى يكون على النظام تحقيقها. أما تصميم النظام فهو عملية استخدام النموذج المنطقى للنظام والأهداف فى عمل توصيف دقيق للتصميم الفعلى (Physical Design) الذى سوف يحقق هذه الأهداف.

وكما سبق أن أوضحنا فإن محلل النظام يعمل إلى جانب العميل على فهم النظام القائم كما يقوم بعمل النموذج المنطقى الذى يحدد بواسطته مواطن الخطأ فى النظام كما يحدد أهداف إنشاء النظام الجديد ثم يقوم بعمل نموذج منطقى للنظام الجديد. ويلاحظ أن هذه الأعمال يمكن تنفيذها دون الحاجة إلى معلومات كبيرة عن تكنولوجيا تشغيل البيانات (Data Processing Technology).

ولكن على الجانب الآخر فإن مصمم النظام يجب أن يكون ملما بأحدث الوسائل التكنولوجية لتشغيل البيانات بما يتيح تصميم النظام بأقل التكاليف مع تحقيق أحسن أداء.. وإذا كان النموذجُ المنطقى للنظام كاملا ودقيقا فإن المصمم لن يحتاج إلى معلومات إضافية عن المجال التطبيقى الذى سوف يعمل فيه النظام.

ويلاحظ مما سبق أننا لم نحاول الدمج بين وظائف التحليل ووظائف تصميم النظام رغم أن هناك بعض الأشخاص الذين يقومون بالجمع بين هذه الوظائف مثل رئيس تحليل النظم (Senior Analyst) والمحلل المبرمج (Programmer / Analyst) وذلك لأن التنظيم الأفضل للعمل خصوصا فى النظم الكبيرة يقتضى فصل هذه الوظائف. وذلك حيث أنه من الصعب على أى شخص أن يكون محيطا إحاطة كاملة بأحد مجالات العمل بما يمكنه من تحديد متطلبات تطوير النظام وفى نفس الوقت يكون ملما بأحدث الوسائل التكنولوجية لتشغيل البيانات بما يمكنه من تصميم النظام. ولكن الوضع الطبيعى أن يكون الشخص الواحد ملما بأحد الفرعين دون الآخر. لذلك يجب أن يكون التمييز بين الوظيفتين واضحا ومحددا حتى يصبح كل منهما مسئولا عن العمل المكلف به.

وليس المقصود بفصل وظيفتى محلل النظم ومصمم النظم منع أى اتصال أو ربط بينهما - كأن يقوم محلل النظم بإنشاء النموذج المنطقى للنظام وتحديد أهداف النظام ثم يترك كل شىء لمصمم النظم - ولكن يجب أن يعمل محلل النظم جنبا إلى جنب مع مصمم النظم. فقد يطلب محلل النظم أن يكون زمن الإستجابة للإستفسارات لايزيد عن قيمة

معينة بينما يجد مصمم النظام أن التصميم الذى سوف يحقق هذا الزمن لن يكون اقتصاديا. وهناك مواقف متعددة تتطلب الإتصال المستمر بين محلل النظام ومصمم النظام. وبصفة عامة فمن المهم جدا أن يعمل محلل النظام مع مصمم النظام مع العميل أو المستخدم فى تقييم البدائل المختلفة لتصميم النظام وتحديد البديل المثالى.

٤٢ - ١ تقييم بدائل التصميم

كما أوضحنا فإن تقييم بدائل التصميم (Physical Alternatives) هو أحد النشاطات أو المهام التى تتطلب اتصالا مباشرا بين محلل النظام ومصمم النظام والمستخدم. حيث أنهم يدرسون بدائل مختلفة للنظام تحقق أهدافا مختلفة مع حسابات مختلفة للتكاليف والوقت. وفى هذه الحالة قد يقوم محلل النظام بإضافة أهداف جديدة للنظام أو إلغاء بعض الأهداف أو تعديل بعض الأهداف وذلك بالتنسيق مع العميل أو المستخدم كما يقوم مصمم النظام بدراسة البدائل المختلفة للتصميم وحساب التكاليف والوقت. وهذا يوضح أهمية مخطط تدفق البيانات (DFD) الذى يكون أداة الإتصال المناسبة بين الأطراف الثلاثة.

ولتوضيح ذلك سوف نقوم بدراسة بعض بدائل تصميم أى نظام معلومات. البديل الأول هو استخدام ملفات الحركة (Transaction Files) فى تجميع البيانات الجديدة من الوحدات الطرفية المختلفة ثم استخدام هذه البيانات فى تحديث بيانات قاعدة البيانات فى نهاية اليوم. وهذا النظام يمتاز بالدقة (Accuracy) وذلك لأن الأخطاء يتم تصحيحها أثناء إدخال بيانات الحركة. كما أنه يوفر فى وقت استخراج التقارير لأن البيانات لاتحتاج إلى تحقيق (Validation) أثناء طباعتها. كما أن النظام يتيح القدرة على الإستفسار عن الموقف الحقيقى للعمل فى نهاية أى يوم. وهذا النظام يستخدم فى أعمال البنوك والمخازن وبعض الأعمال الأخرى.

والبديل الثانى هو استخدام الإدخال المباشر للبيانات (On-Line) وهذا يعنى أن المستخدم يقوم بإخال بيانات الحركة فيتم بواسطتها تحديث بيانات قاعدة البيانات مباشرة. وهذا النوع من النظم يكون مكلفا نسبيا سواء فى مكونات الأجهزة (Hardware) أو فى صيانة وتطوير البرامج. ولكن له فوائد متعددة فالبيانات يتم تصحيحها أثناء إدخالها كما أن قاعدة البيانات تعطى فى أى وقت بيانات سليمة وحديثة. فمثلا فى نظام حجز تذاكر الطيران يمكن الإستفسار عن الأماكن الخالية فى الطائرة فى أى وقت بعد آخر حجز للأماكن حتى وإن كان هذا الحجز لم يمض عليه ثوان معدودة.

والبديل الثالث هو استخدام النظم الموزعة (Distributed Systems) الذى يعتمد على

حاسب متوسط (Minicomputer) ووحدات طرفية خاصة (Dedicated Terminals) تحتوى على إمكانيات إدخال البيانات وتصحيحها. ويتم من خلال الوحدات الطرفية إدخال بيانات الحركة واستخراج التقارير كما تقوم بإرسال البيانات فى الليل إلى الحاسب المركزى ليستخدمها فى تحديث قاعدة البيانات.

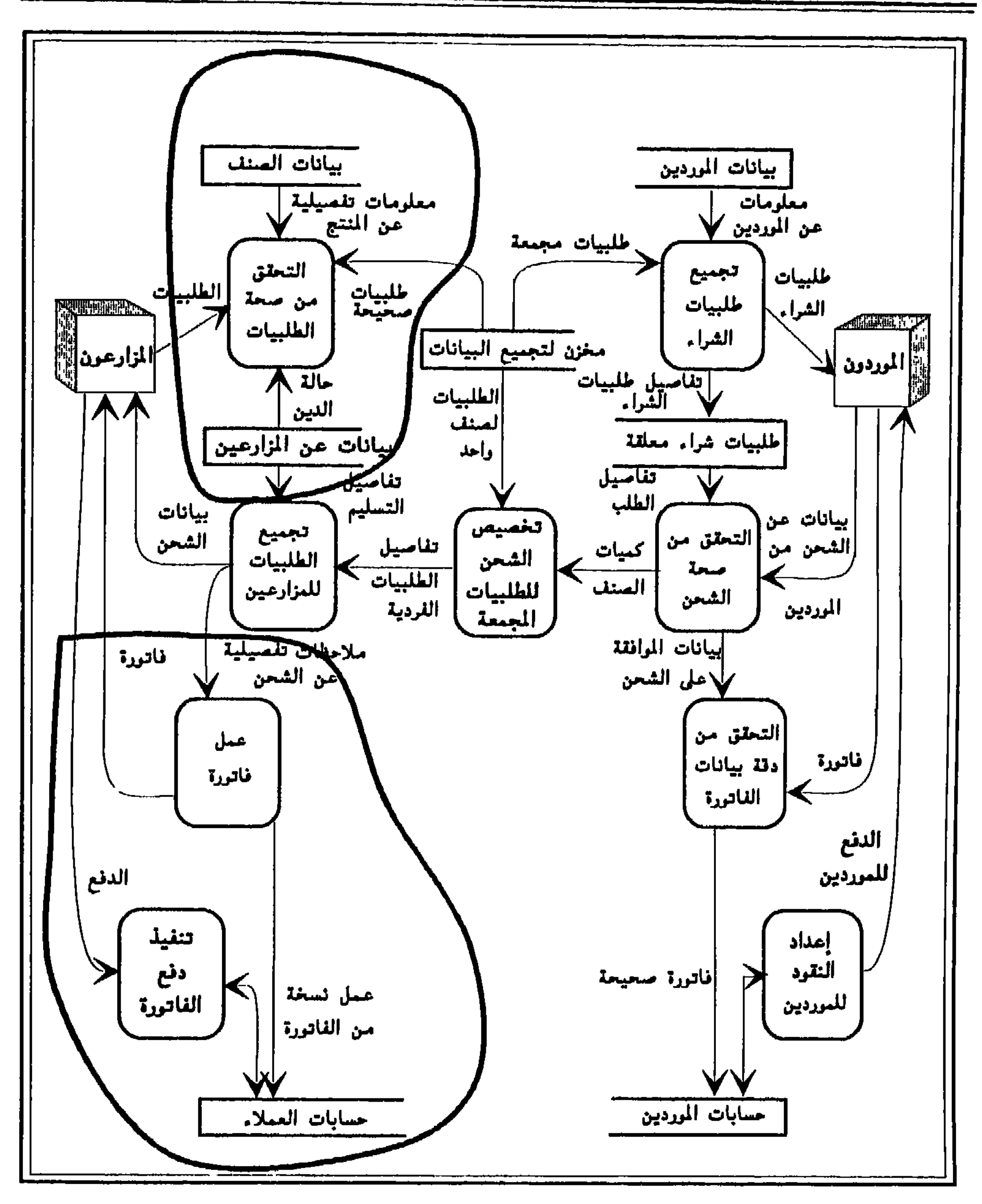
والبديل الرابع هو إنشاء نظام يدوى محسن فقد يوضح مخطط تدفق البيانات أن هناك تكرارا وازدواجية يمكن التخلص منها. وقد يستنتج محلل النظم أن تحقيق متطلبات المستخدم لايقتضى ميكنة النظام ولكن يتطلب بعض التغييرات التنظيمية والوظيفية أو استبدال بعض الأشخاص.

والبديل الخامس هو إنشاء نظام يمزج بين النظام اليدوى والنظام الآلى. وهذا يعنى أن المحلل والمصمم قد يجدان أن بعض الوظائف يمكن تنفيذها بالنظام اليدوى وبعض الوظائف الأخرى تكون ميكنتها أكثر جدوى. ويمكن تحديد هذه الوظائف من خلال مخطط تدفق البيانات. أنظر شكل (٤٢ - ١)

ويلاحظ من الشكل أنه تم رسم حدود (Boundaries) حول الوظائف المطلوب ميكنتها.

٤٢ - ٢ التصميم القابل للتعديل (Changeable Design)

من المشاكل التى تقابل مصممى ومبرمجى النظم عند تصميم النظام هى عدم قدرة النظام على الإستجابة السريعة لاى تعديلات يطلبها المستخدم. والمشكلة الثانية هى عدم القدرة على ربط أجزاء النظام ببعضها أو التأكد من تحقيق التكامل (Integrity) بينها. وأدت دراسة هذه المشاكل إلى اكتشاف وسائل البرمجة التركيبية (Structural Programming). والبرمجة التركيبية هى سلسلة من البرامج الفرعية التى لكل منها مدخل واحد ومخرج واحد (Single entry / Single exit) ويسمى أيضا الصندوق الأسود (Black Box). وكل برنامج فرعى من هذه البرامج يحتوى على التراكيب الأساسية للبرمجة وهى التسلسل المنطقى (Logical Sequence) والتفرع (Branching) والتكرار (Looping). وكلما تم تقليل عدد التراكيب الأساسية للبرمجة داخل البرنامج الفرعى كلما كان البرنامج الفرعى أسهل فى إنشائه وفهمه وتعديله. والشكل (٤٢ - ٢) يوضح أحد البرامج التركيبية المكتوبة بلغة (Pseudo-COBOL). ويلاحظ أن البرنامج يحتوى على التراكيب الأساسية للبرمجة السابق شرحها.



شكل (٤٢ - ١)

والجملة الأولى فى البرنامج (PERFORM INITIALIZATION) توضح أن هناك برنامجا فرعيا يسمى (INITIALIZATION) فى مكان ما من البرنامج. وكلمة (PERFORM) تعنى نقل التحكم (Control) إلى البرنامج الفرعى

(INITIALIZATION) وتنفيذ الجمل المكتوبة فيه ثم العودة إلى الجملة التالية فى البرنامج الرئيسى. فإذا كان البرنامج (INITIALIZATION) مكتوبا بالطريقة التركيبية (Single entry/Single exit) تصبح الجملة (PERFORM INITIALIZATION) مجرد جملة واحدة فى البرنامج الرئيسى مثل الجمل الأخرى ولا يترتب على تنفيذها أى مشاكل.

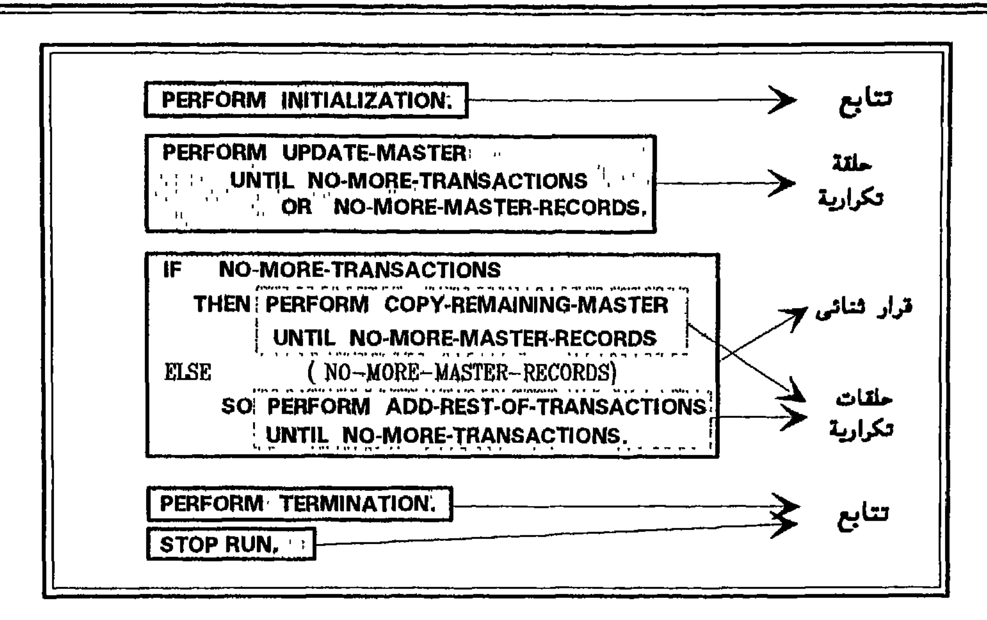
```
PERFORM INITIALIZATION.
PERFORM UPDATE-MASTER
      UNTIL NO-MORE-TRANSACTIONS
            OR  NO-MORE-MASTER-RECORDS.
IF    NO-MORE-TRANSACTIONS
   THEN PERFORM COPY-REMAINING-MASTER
        UNTIL NO-MORE-MASTER-RECORDS
ELSE      ( NO-MORE-MASTER-RECORDS)
     SO PERFORM ADD-REST-OF-TRANSACTIONS
      UNTIL NO-MORE-TRANSACTIONS.
PERFORM TERMINATION.
STOP RUN.
```

شكل (٤٢ - ٢)

والجملة الثانية (PERFORM UPDATE-MASTER UNTIL) هى حلقة تكرارية (Loop). وهى تعنى أن هناك برنامجا فرعيا (UPDATE-MASTER) يتم تنفيذه عدة مرات حتى تنتهى بيانات الحركة أو ينتهى الملف الرئيسى (Master File).

وكما قلنا سابقا فإن البرنامج (UPDATE-MASTER) إذا كان مكتوبا بطريقة تركيبية (Single entry/Single exit) فسوف يتم تنفيذه والرجوع إلى الجملة التالية دون حدوث أى مشاكل.

والجملة الثالثة هى (IF ... THEN ... ELSE) وهى تؤدى إلى التفرع إلى البرامج الموضحة بناء على تحقق الشرط المكتوب بعد (IF). وأخيرا هناك الجمل المتوالية (Sequence) التى تقوم بإيقاف البرنامج. أنظر شكل (٤٢ - ٣)

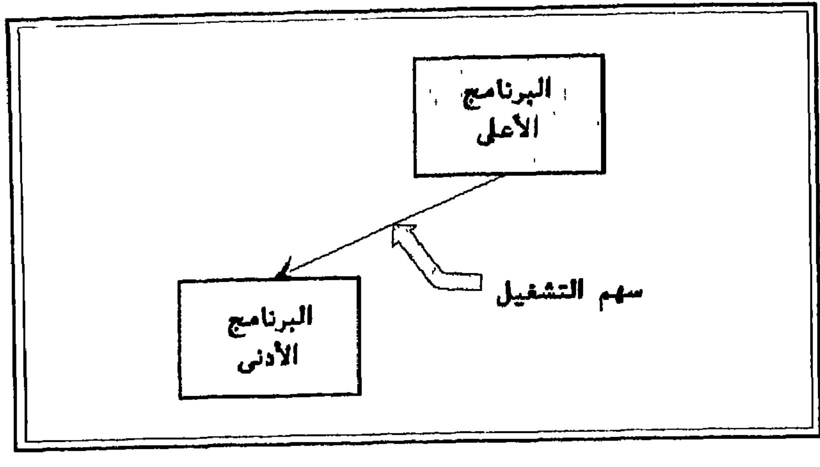


شكل (٤٢ - ٣)

مما سبق يلاحظ أن البرنامج الرئيسى يحتوى على برامج فرعية ذات مستوى أقل (Low Level Modules). كما أن البرامج الفرعية أيضا قد تحتوى على برامج فرعية ذات مستوى أقل. لذلك تسمى هذه الطريقة فى البرمجة البرمجة التركيبية الهرمية (Hierarchical Structural Programming). وهى تمتاز بسهولة الإختبار والتصحيح حيث يمكن اختبار التسلسل المنطقى لكل برنامج فرعى على حدة. ولكن البرنامج بهذه الصورة لا يوضح المستويات المختلفة للبرامج الفرعية. وإذا كان البرنامج كبيرا تصبح عملية تحديد البرامج الفرعية ومستوى كل برنامج عملية صعبة جدا. لذلك تستخدم وسيلة أو أداة أخرى تسمى مخطط التركيب (Structure Chart) وفى هذا المخطط يتم تمثيل كل برنامج بمستطيل بحيث يقوم البرنامج الأعلى بتشغيل البرنامج الأقل منه مستوى. لذلك تكون المستطيلات موصلة بأسهم تتجه من المستطيل الأعلى إلى المستطيل الأدنى وتسمى هذه الأسهم أسهم التشغيل (Invocation Arrows). وهذه الأسهم تختلف عن أسهم البيانات فى خريطة تدفق البيانات مثلا. فهى أسهم توضح حدثا ذى إتجاهين (Two-way event) حيث يقوم البرنامج الأعلى أولا بتشغيل البرنامج الأدنى. وعند الإنتهاء من تنفيذ البرنامج الأدنى يعود التحكم مرة ثانية إلى البرنامج الأعلى. أنظر شكل (٤٢ - ٤)

والقدرة على رسم مخططات التركيب والتفكير فى البرامج بهذه الصورة يكون فى منتهى الأهمية. لأنه يساعد على النظر إلى المشكلة بالطريقة الهرمية وليس بطريقة مخطط

التدفق (Flow Chart) كتسلسل أحداث. وهذا يؤدى إلى سهولة كتابة البرنامج التركيبى الهرمى بناء على هذه المخططات. كما أن مخططات التركيب هى الوسيلة أو الأداة التى تساعد على تصميم النظم القابلة للتعديل (Changeable Systems).



شكل (٤٢ - ٤)

ولكن ما المقصود بالنظم القابلة للتعديل ؟

المقصود بالنظم القابلة للتعديل هى النظم التى يمكن تعديل جزء من مكوناتها بسهولة ودون تأثير هذا التعديل على باقى مكونات النظام. وهناك اتفاق عام على أن أحسن النظم قابلية للتعديل هى تلك النظم المكونة من برامج صغيرة مستقلة (Independent) بقدر الإمكان عن بعضها.

والبرامج الفرعية (Modules) داخل البرنامج يمكن تصنيفها إلى الرئيس (Boss) والرئيس الأقل (Sub-boss) وهكذا حتى الوصول إلى البرامج العاملة (Workers) التى تؤدى وظيفة محددة. وعندما نقول أن البرامج الفرعية يجب أن تكون صغيرة فالمقصود بذلك أنه يسهل قراءتها واستنتاج وظيفتها وبالتالى تعديلها. ورغم أن هذه البرامج يتفاوت حجمها إلا أنه يمكن إعتبار (٥٠ - ١٠٠) سطر حجما مثاليا للبرامج العاملة (Worker Modules).

والشرط الثانى لكى يصبح النظام قابلا للتعديل (Changeable) هو أن تكون البرامج الفرعية مستقلة (Independent). وليس المقصود باستقلال البرامج أن تكون مستقلة تماما عن بعضها وإلا لن يكون هناك نظام. ولكن المقصود أن تكون الروابط بين

هذه البرامج مصممة بعناية فائقة وأن تكون تحت السيطرة. وإذا كانت الروابط بين البرامج غير واضحة فإن هذا يؤدي إلى ما يسمى بالتأثير المتموج (Ripple Effect) والذى يعنى أن أى تعديل فى أحد البرامج يسبب مشاكل فى عدة برامج أخرى بسبب الروابط بينها.

فمثلا لو فرضنا أن العميل طلب تعديلا يؤثر فى البرنامج الفرعى (A) فإن المبرمج يقوم بتعديل البرنامج لإدخال هذا التعديل. ولكن عند تشغيل النظام يفاجأ بظهور مشكلة فى البرنامج (B). فيقوم المبرمج بتحديد المشكلة وبالتالى يعدل البرنامج (B). ولكن التعديل فى البرنامج (B) يؤثر فى البرنامج (C). وتعديل البرنامج (C) يؤثر فى البرنامج (D) وهكذا. ويلاحظ هنا أن التعديل فى أحد البرامج ينتشر خلال النظام وينتقل التأثير من برنامج إلى آخر. فإذا كانت الروابط بين البرامج الفرعية متعددة يصبح تعديل النظام شبه مستحيل.

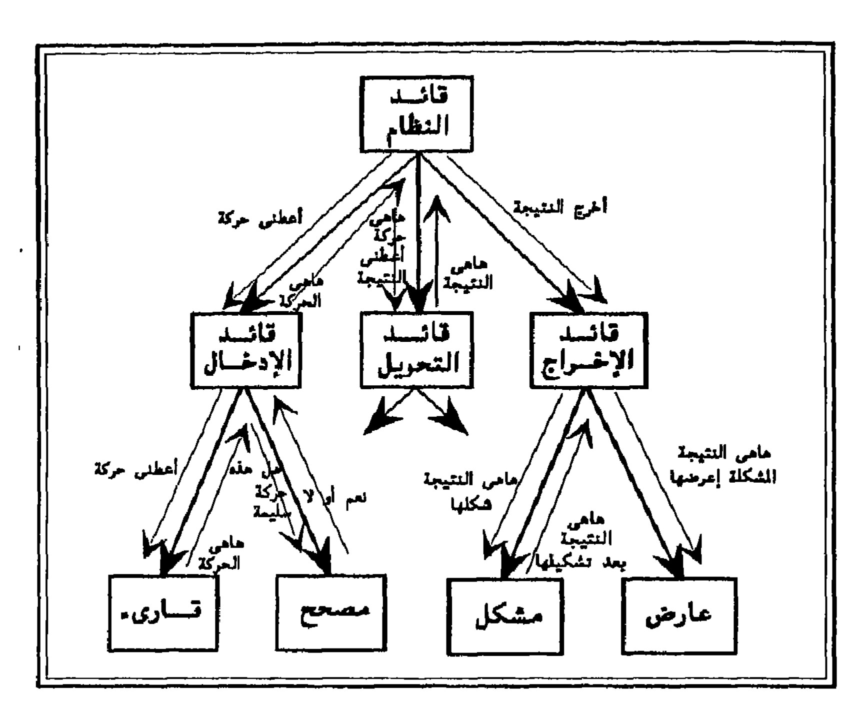
لذلك فإن وظيفة مصمم النظام هى تصميم البرامج وتحديد الروابط بينها لتقليل أثر التأثير المتموج (Ripple Effect).

٤٢ - ٣ التصميم الهرمى التركيبى للنظام

كما أوضحنا فإن التصميم الهرمى التركيبى للنظام هو التصميم الذى يحقق النظام القابل للتعديل (Changeable System). ويمكن تشبيه التصميم بهذه الطريقة بتنظيم الوحدات العسكرية حيث أن كل برنامج فرعى له وظيفة محددة لايبدأ فى تنفيذها إلا عندما تصدر إليه التعليمات من البرنامج الأعلى. كما أن كل برنامج يتصل فقط ببرنامج قائد واحد بالإضافة إلى واحد أو أكثر من البرامج الأدنى. والشكل (٤٢ - ٥) يوضح التصميم الهرمى التركيبى للنظام.

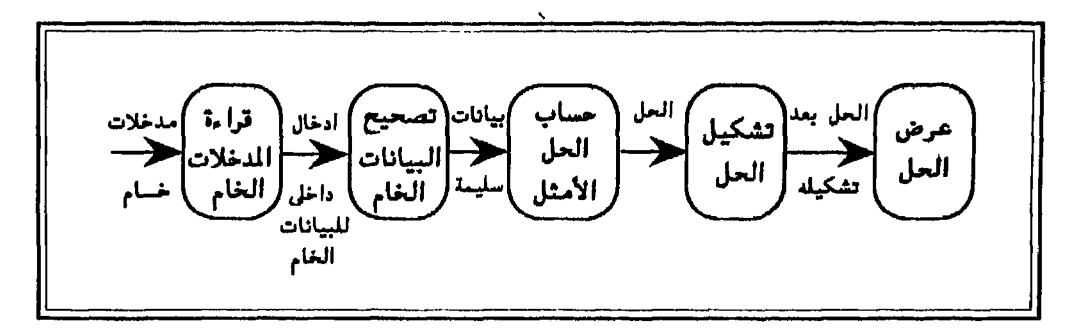
ويلاحظ من هذا الشكل أن برنامج قائد النظام (System Commander) يقوم بتشغيل قائد المدخلات (Input Commander) عن طريق الأمر (Get me a good transaction). ويقوم قائد الإدخال (Input Commander) بتشغيل برنامج القارىء (Reader) باستخدام الأمر (Get me a transaction) فيقوم القارىء (Reader) بإحضار بيانات الحركة(Transaction) إذا كانت متاحة ثم يعيد التحكم إلى قائده مرة أخرى قائلا(Here is a transaction) أو(There is no more transactions) فى حالة عدم وجود بيانات حركة. ثم يقوم قائد المدخلات (Input Commander) بتشغيل برنامج المصحح (Editor) ويرسل إليه بيانات الحركة ويسأله (Is this transaction good?) فيقوم برنامج المصحح (Editor) بأداء وظيفته ويرد على

قائد المدخلات. وإذا كانت بيانات الحركة غير سليمة فإن قائد المدخلات يتخذ قراره إما بتصحيح هذه البيانات أو بإلغاء هذه الحركة وطلب حركة جديدة من برنامج القارىء (Reader). وإذا كانت بيانات الحركة سليمة يقوم قائد المدخلات (Input Commander) بإعادة التحكم إلى قائد النظام (System Commander) الذى ينقل التحكم إلى قائد التحويل (Transform Commander) ويقوم قائد التحويل بتشغيل الوحدات التابعة له لتحويل البيانات إلى الصورة الملائمة للإخراج ويعيد بيانات الحركة إلى قائد النظام فيقوم بتحويلها إلى قائد المخرجات (Output) لعرضها.



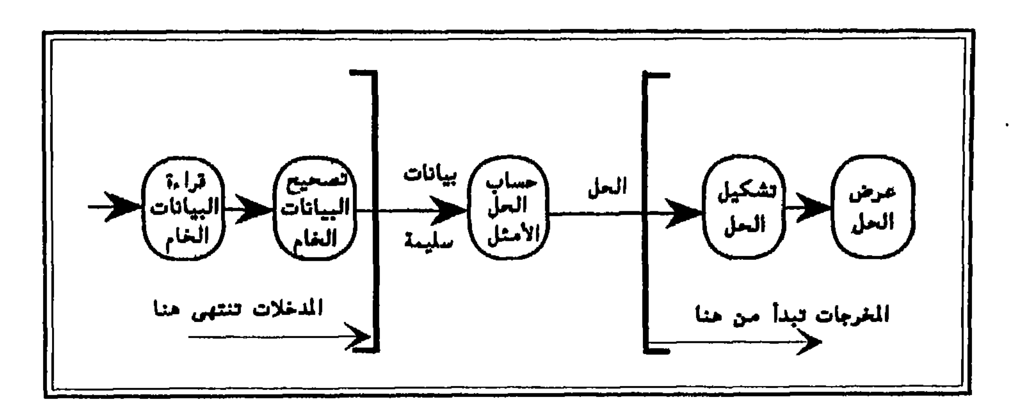
شكل (٤٢ - ٥)

وهذا النوع من النظم الذى يحتوى على فرع خاص بالإدخال (Input Leg) يقوم بكل وظائف الإدخال ، وفرع خاص بالتحويل (Transform Leg) يقوم بكل عمليات تحويل المدخلات إلى مخرجات ، وفرع خاص بالإخراج (Output Leg) يقوم بكل عمليات الإخراج. هذا النوع يسمى نظم التحويل المركزى (Transform Centered Systems). ويمكن الحصول على التركيب الهرمى الخاص بها عن طريق خريطة تدفق البيانات (DFD) التى تكون مثلا كالموضح بالشكل (٤٢ - ٦).



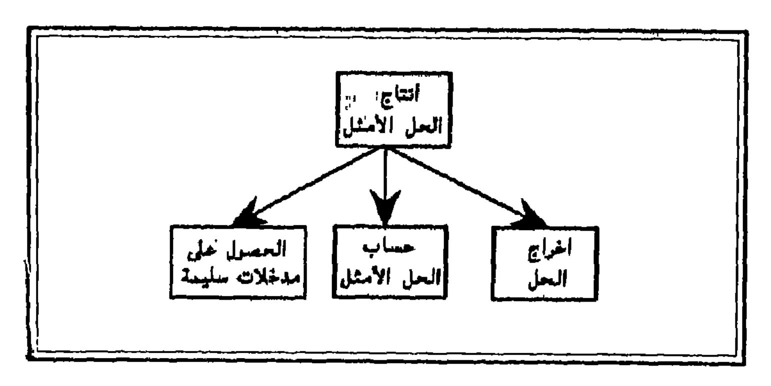
شكل (٤٢ - ٦)

ولرسم التركيب الهرمى من خريطة تدفق البيانات الموضحة بالشكل نبدأ بالمدخلات الخام (Raw Input) ونتتبعها حتى نصل إلى النقطة التى لايمكن اعتبارها مدخلات. وبنفس الطريقة نبدأ بالمخرجات النهائية ونتتبعها حتى نصل إلى النقطة التى لايمكن اعتبارها مخرجات. والشكل (٤٢ - ٧) يوضح خريطة التدفق بعد تحديد هذه النقط بواسطة أقواس مربعة كبيرة.



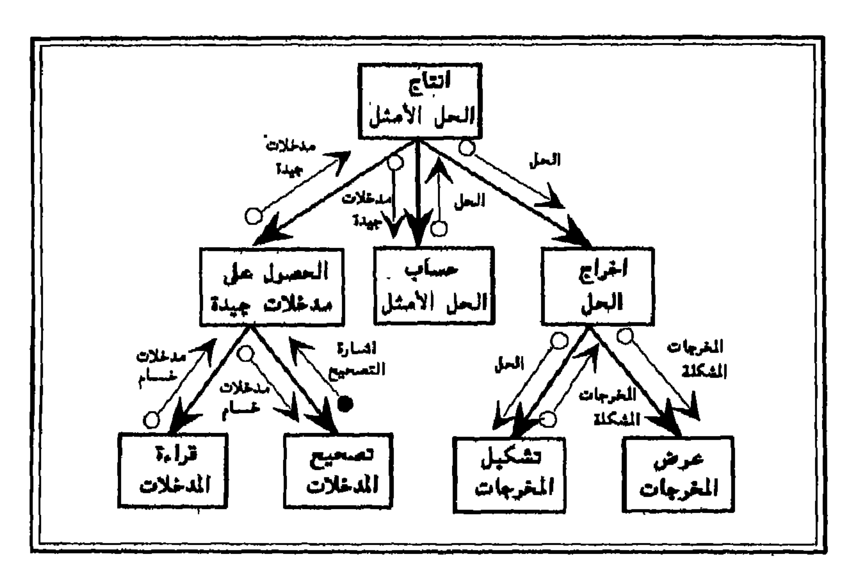
شكل (٤٢ - ٧)

وبعد تحديد هذه النقط يتم إنشاء تصميم التحويل المركزى (Transform Centered Design) موضحا النظام الذى يقوم باستدعاء المدخلات وتحويلها إلى مخرجات ثم عرض هذه المخرجات. والشكل (٤٢ - ٨) يوضح المستوى الأول من هذا التصميم.



شكل (٤٢ - ٨)

والشكل (٤٢ - ٩) يوضح مستوى أقل (مستوى تفاصيل أكبر) ويلاحظ أن هناك أسهم لها دائرة مفتوحة وهى تعبر عن حركة البيانات. وهناك أسهم لها دائرة مملوءة وهى تعبر عن نقل التحكم من برنامج إلى برنامج آخر.

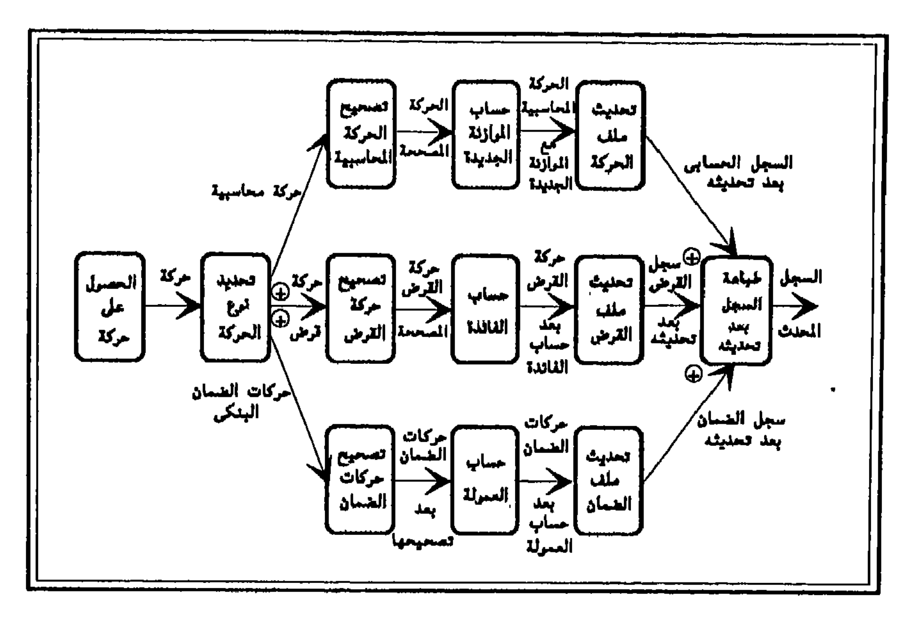


شكل (٤٢ - ٩)

٤٢ - ٤ مثال عملى على التصميم الهرمى التركيبى

كما سبق أن أوضحنا فإن التصميم الهرمى التركيبى للنظام يتم الحصول عليه من خريطة تدفق البيانات (DFD). كما أن نظام التحويل المركزى

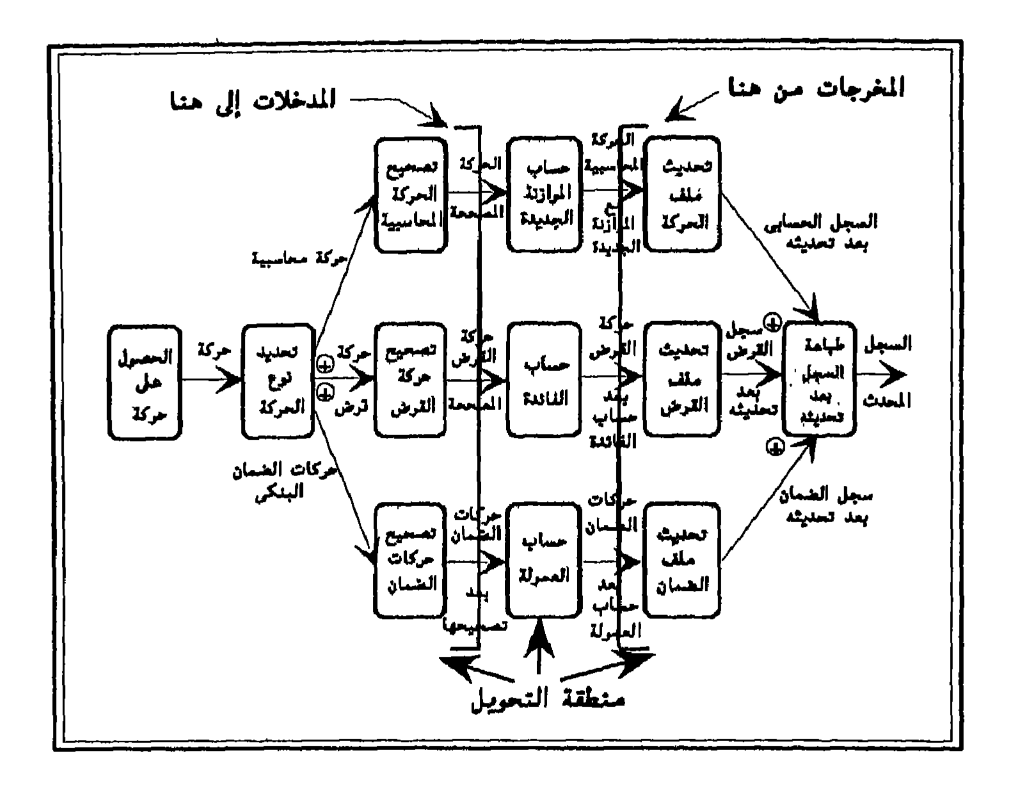
(Transform-Centered System) يقتضى فصل جميع عمليات الإدخال (Input) وعمليات التحويل (Transform) وعمليات الإخراج داخل خريطة تدفق البيانات(DFD). وهناك نظم تتضمن بيانات حركة (Transactions) تتفرع فى مسارات مختلفة وتمر بنفس عمليات الإدخال والتحويل والإخراج. ومن هذه النظم مثلا النظم المالية والمحاسبية (Accounting Systems). والشكل (٤٢ - ١٠) يوضح أحد النظم المالية ويظهر فيه بوضوح أن هناك ثلاثة فروع لبيانات الحركة أحدها خاص ببيانات حركة الحسابات (Account Trans) والثانى خاص ببيانات حركة القروض (Loan Trans) والثالث خاص ببيانات حركة السندات (Securities Trans).



شكل (٤٢ - ١٠)

ويلاحظ من الشكل استخدام دوائر داخلها علامة (+) لتوضح أن بيانات أى حركة تمر خلال واحد فقط من الفروع الثلاثة أى أن العلاقة المنطقية بين الفروع الثلاثة هى (OR). أما إذا كانت العلاقة المنطقية هى (AND) فإن هذا يعنى أن بيانات أى حركة تمر خلال الفروع الثلاثة معا.

وكما سبق أن أوضحنا فإن أول خطوة هى توضيح الجزء من خريطة تدفق البيانات الذى يمكن إعتباره مدخلات (Input) والجزء الذى يمكن إعتباره تحويل (Transform) والجزء الذى يمكن إعتباره مخرجات (Output). وهذه الأجزاء الثلاثة يتم توضيحها بالأقواس المربعة كما يتضح من الشكل (٤٢ - ١١)



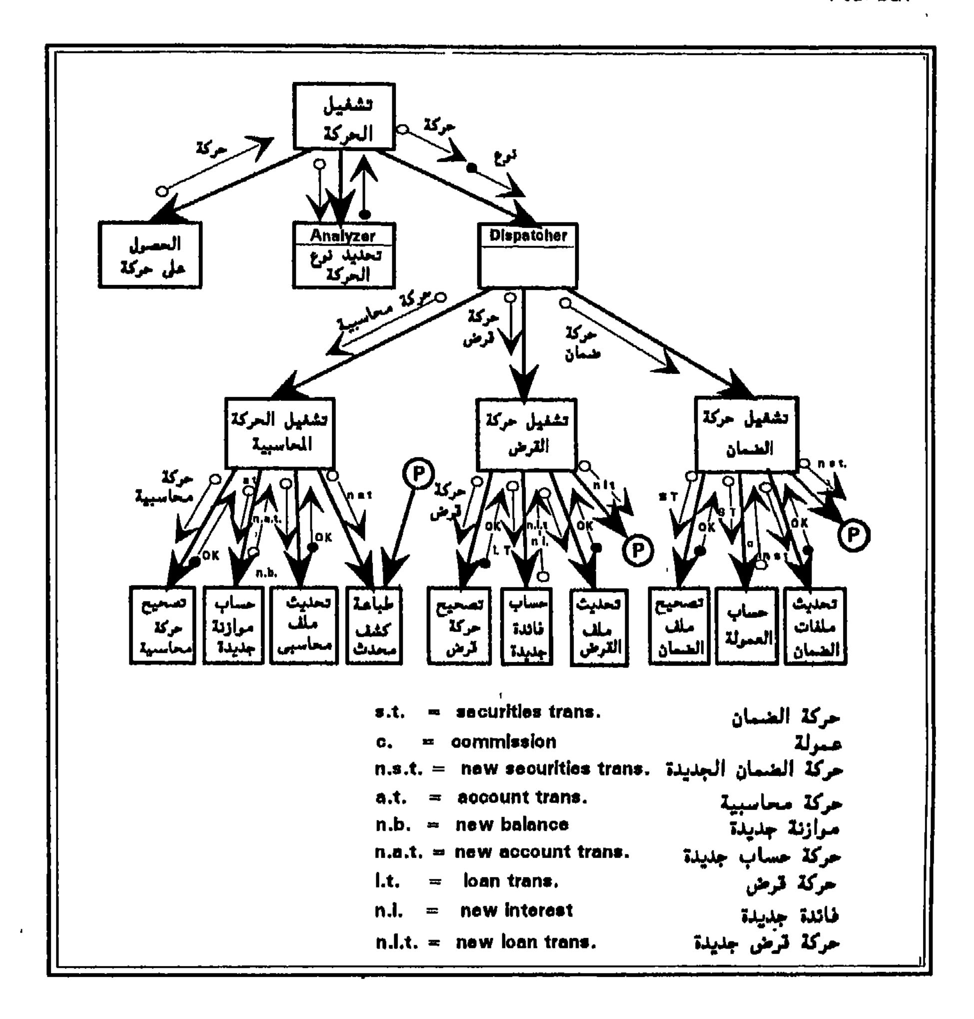
شكل (٤٢ - ١١)

ولكى يتم الحصول على الشكل التركيبى الهرمى المقابل لهذا النموذج نحتاج أولا إلى برنامج فرعى يقوم بتشغيل ثلاثة برامج فرعية أخرى الأول يقوم بالحصول على بيانات الحركة (Get a Trans.) ، والثانى يقوم بتحليل هذه البيانات وتحديد نوع كل حركة (Analyzer) ، والثالث يقوم بتوزيع هذه الحركة على أحد الفروع الثلاثة(Dispatcher). أنظر شكل (٤٢ - ١٢).

ويلاحظ من الشكل أن هناك تدفق بيانات (Data Flow) وكذلك تدفق تحكم (Control Flow) بين البرامج. وإذا ربطنا بين هذا الشكل وبين خريطة تدفق البيانات نلاحظ الآتى :

١ - كل عملية منطقية داخل خريطة تدفق البيانات (أى المكتوبة داخل مستطيلات مستديرة الأركان) تتحول فى التصميم التركيبى الهرمى إلى برنامج عامل (Worker Module).

٢ - لايتم تحويل أى عملية منطقية داخل خريطة البيانات إلى برنامج مدير (Manager Module). ولكن البرنامج المدير يتم وضعه للتحكم فى البرامج العاملة.



شكل (٤٢ - ١٢)

ويمكن تلخيص خطوات تصميم النظام التركيبى الهرمى كالآتى :

١ - يتم أولا رسم خريطة تدفق البيانات لتوضيح الوظائف التفصيلية للنظام.

٢ - يتم تحديد الجزء الخاص بالمدخلات (Input) والجزء الخاص بالتحويل (Transform) والجزء الخاص بالمخرجات (Output).

٣ - يتم إنشاء المخطط التركيبي الذى يتم فيه تحويل كل وظيفة موجودة فى خريطة تدفق البيانات إلى البرنامج العامل المقابل (Worker Module). كما يتم إنشاء برامج التحكم (Manager Modules) التى تتحكم فى البرامج العاملة.

٤ - يتم تحديد تدفق البيانات (Data Flow) وتدفق التحكم (Control Flow) بين البرامج.



الفصل الثالث والأربعون

نحو ميكنة أكبر
لتطوير النظم



٤٣ - ١ مقدمـــــة

ميكنة تطوير النظم أصبحت حاليا هى الشغل الشاغل للباحثين فى مجال الحاسب. واصبح هناك إتجاه مستمر نحو مزيد من الميكنة لأنها تؤدىَ إلى إسراع عملية التطوير وكذلك تقليل التكلفة. وهناك هدف آخر أساسى للميكنة وهو مساعدة العقل البشرى الذى يكون عادة محدود الإمكانيات بالنسبة للوظائف الحسابية المعقدة أو تلك التى تتطلب تخزين كمية كبيرة من البيانات وإسترجاعها.

ويقوم محلل النظام الذى يستخدم أدوات التطوير الحديثة بإنشاء التصميمات المطلوبة فى محطة خاصة (Workstation) تحتوى على حاسب بالإضافة إلى البرمجيات المناسبة التى تساعده على تصميم وتعديل المخططات والجداول (Diagrams) وهذه المخططات سوف تكون من الدقة بحيث يمكن إعتبارها لغة لاتختلف عن لغات البرمجة من حيث تحقيق الإتصال بين المستخدم والحاسب ثم يتم تحويل هذه اللغة إلى كود منفذ (Executable Code). وهذه الأدوات الحديثة فى مجموعها هى مايطلق عليه أدوات هندسة البرامج (CASE Tools) والتى يتم شرح أهم خصائصها فى هذا الفصل كما يتم شرح أحد المنهجيات التى تستخدم هذه الأدوات فى الفصل التالى.

٤٣ - ٢ مشاكل المتطلبات

من أكبر المشاكل التى تواجه تطوير النظم حاليا هو تحديد متطلبات النظم الكبيرة المعقدة. وعادة يقوم محللو النظم الذين يستخدمون الطرق التقليدية بتحديد المتطلبات التى تحتوى على العديد من الأخطاء والأجزاء غير المطابقة أو المحذوفة. وهناك منهجيات متعددة لتحديد المتطلبات مثل (Yourdon) و (Constantine) و (De Macro) و (Jakson) و (Warnier and Orr) وهناك منهجيات سبق شرحها مثل خرائط هيبو (HIPO Charts) ومنهجية هندسة البرامج (SREM) وخرائط تدفق البيانات (DFD). وقد حاولت هذه المنهجيات تحسين كفاءة تحديد المتطلبات ولكنها لم تحقق الهدف تماما كما أنها لم تكن دقيقة أو قابلة للقياس الرياضى (Rigorous) وعلاوة على ذلك فإنها لم تكن تولد الكود آليا. ثم ظهرت منهجيات أخرى تستخدم أدوات هندسة البرامج (CASE Tools) مثل منهجية (HOS) التى سوف يتم شرحها فى الفصل التالى وهى تتخلص إلى حد كبير من عيوب المنهجيات السابقة كما أنها تولد الكود آليا.

٤٣ - ٣ لغات تحديد المتطلبات

فى السبعينات كانت هناك عدة محاولات للوصول إلى لغة لتحديد المتطلبات. ولكن لم تصل هذه اللغات أو أى لغات أخرى حديثة إلى دقة لغات البرمجة. وحتى تصل أى لغة متطلبات إلى دقة لغات البرمجة فإنها يجب أولا أن تكون قابلة لتوليد الكود آليا. فإن ذلك يتيح اختبار هذه اللغة ومقارنة المخرجات بالمتطلبات.

وتختلف لغة المتطلبات عن لغة البرمجة فى أنها لاتتكون من تعليمات مثل لغة البرمجة ولكنها تكون على هيئة رسومات ومخططات. ولكنها تعطى نفس تأثير اللغة من حيث تحقيق إتصال المستخدم بالحاسب. ولأن حاسبات اليوم أصبحت تتمتع بخصائص جرافيكية عالية فقد أصبح فى الإمكان تصيم أشكال ومخططات تعبر عن المتطلبات ويمكن اختبارها آليا وتنقيحها ثم تحويلها إلى كود منفذ (Executable Code). وكلما أراد المستخدم تعديل المتطلبات فإن التصميم يمكن ضبطه وإعادة توليد البرامج آليا. لذلك فإن لغة المتطلبات يجب أن تكون سهلة الفهم حتى يستطيع المستخدم دراسة هذه المتطلبات ومناقشة محلل النظم فى مدى توافقها مع المتطلبات الفعلية.

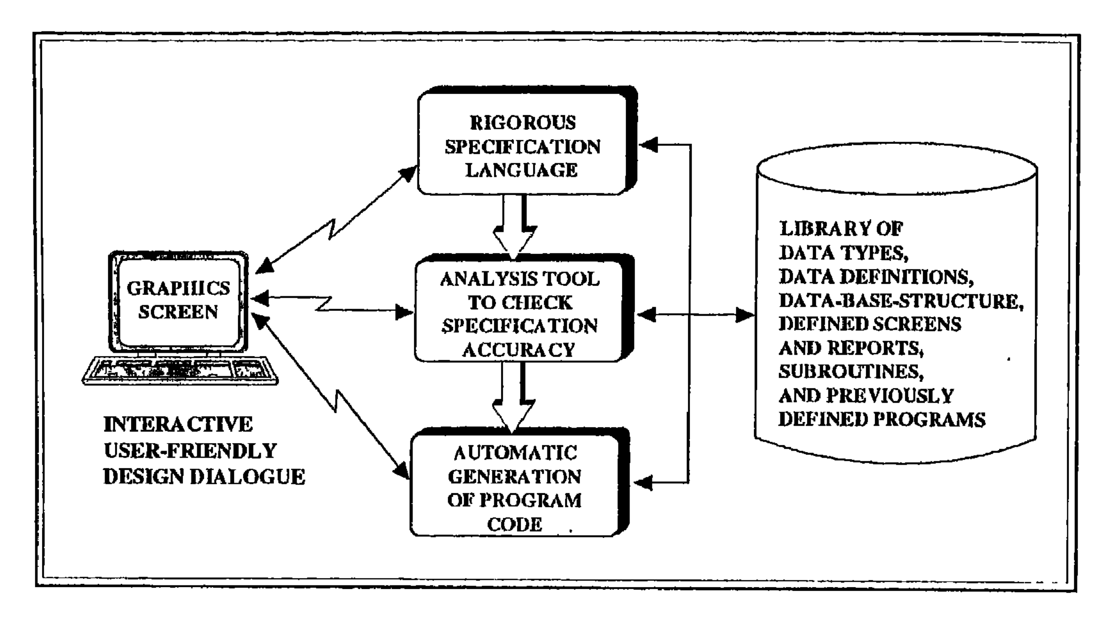
ومن أمثلة لغات تحديد المتطلبات مولد التقارير (Report Generator) حيث يستطيع المستخدم من خلاله تحديد المعلومات المطلوب عرضها فى التقرير. ولكن مولد التقارير هو فى الواقع جزء صغير جدا من أدوات الـ (CASE) التى تشمل تحديد كل متطلبات المستخدم من النظم بالإضافة إلى القدرة على اختبار دقة توصيف هذه المتطلبات ودقة البيانات المستخدمة. وهذه الأدوات تستطيع كشف أى غموض أو عدم توافق أو حذف فى المتطلبات وذلك لأنها تكون متطلبات حاسبية (Computable) أى قابلة للتشغيل على الحاسب. والعقل الإنسانى لايستطيع إكتشاف أى غموض أو نقص فى توصيف المتطلبات فى النظم المعقدة كما أن فريقا من العقول الإنسانية يؤدى إلى نتيجة أسوأ لأنهم فى العادة يبنون أجزاء غير متوافقة فيما بينها.

٤٣ - ٤ ميكنة التصميم

تصميم النظام لن يكون أبدا آليا تماما لأن الإنسان عادة يتميز بالقدرة على الإبتكار والإبداع. وسوف يحتاج الإنسان باستمرار إلى رسم أشكال كروكية على الورق ومناقشتها. لذلك فإن أدوات التصميم يجب أن تتيح رسم الأشكال البسيطة اللازمة. والشكل (٤٣ - ١) يوضح العناصر الرئيسية فى تصميم النظام. ويوضح الشكل وجود

لغة توصيف المتطلبات التى تتميز بالدقة الشديدة (Rigorous) كما تتميز بتفاعلها مع المستخدم (Interactive) واستخدام القوائم السهلة. وهذا التصميم المعتمد على الحاسب يسهل تحليله (Analyze) لاختبار دقته وشموله كما يتضح من الشكل.

وبعد تحليل التصميم واختباره يتم توليد الكود. وهناك أدوات تولد الكود بلغات البرمجة التقليدية وهذا يوفر برامج نقاله (Portable) يمكن تشغيلها على أجهزة مختلفة. كما يمكن توليد كود بلغة الآلة (Machine Code) وهذا يكون أكثر كفاءة لأنه يوفر الخطوة الخاصة بالترجمة (Compilation).



شكل (٤٣ - ١)

وتقوم عملية تحديد المتطلبات على إنشاء مخطط للنظام يمكن تحليله إلى تفاصيل أكثر. ويكون على أفراد النظام الإشتراك فى مستويات التفصيل المختلفة. ومن المفيد أن يستخدم الجميع نفس المخطط. وفى أعلى مستوى (Highest Level) للمخطط تظهر البلوكات التى توضح المتطلبات الرئيسية للنظام. وفى أدنى مستوى (Lowest Level) تظهر البلوكات التى تحتوى على تفاصيل كافية لتوليد الكود. ويتم تصميم كل جزء فى النظام منفصلا على أن يتم ربط الأجزاء ببعضها. وعند إجراء أى تعديل للنظام فإن هذا التعديل يتم على أدنى مستوى ويظهر أثره على المستوى الأعلى وبالتالى يمكن اختبار النظام من خلال المستوى الأعلى فقط.

٤٣ - ٥ تكامل التصميم

فى نظم التصميم التقليدية يتم تحديد المتطلبات بطريقة معينة (باللغة الانجليزية مثلا) ويتم توصيف المتطلبات بطريقة أخرى كما يتم تطبيق هذا التصميم بطريقة أخرى (لغة البرمجة). وعند ترجمة المتطلبات للحصول على توصيف المتطلبات تكون هناك بعض الأخطاء. وعند ترجمة التوصيف إلى كود تكون هناك بعض الأخطاء الأخرى. وعند حدوث أى تغيير فى توصيف المتطلبات فإن هذا التغيير لن يظهر على المتطلبات الفعلية كما أن أى تعديل فى الكود لن يظهر أثره على توصيف المتطلبات. كما أن أى تعديل فى النظام لن يظهر أثره على وثائق النظام (Documentation) وعادة يتم اختبار النظام عن طريق اختبار وثائق المتطلبات بطريقة يدوية تستهلك وقتا طويلا.

أما أدوات الـ (CASE) فإنها تتغلب على هذه المشاكل عن طريق مجموعة من الوسائل التخطيطية التى يتم من خلالها تمثيل المتطلبات والتوصيف والتفاصيل. حيث يتم تحليل المتطلبات للحصول على المواصفات (Specifications) التى يتم تحليلها إلى تفاصيل أكثر حتى يتم توليد الكود فى النهاية. وعند حدوث أى تعديل فى المستويات الأدنى فإن هذا التعديل ينعكس على المستويات الأعلى وبالتالى يكون هناك تكامل تركيبى بين المتطلبات والمواصفات والكود. كما أن وثائق النظام يتم تحديثها آليا مع كل تغيير فى النظام.

٤٣ - ٦ الأدوات المدققة رياضيا

لتحقيق الدقة المطلوبة للنظام فإن بعض أدوات الـ (CASE) يتم بناؤها على أسس رياضية (Mathematically Based). فمثلا فى البرنامج (USE. IT) المستخدم مع منهجية (HOS) فإن كل أجزاء النظام يتم بناؤها على أسس رياضية كما سيتم الإيضاح فى الفصل التالى. وهذا يؤدى إلى بناء هياكل مدققة رياضيا. وكل جزء فى النظام يتم بناؤه من عناصر أولية (Primitives) مبنية على أسس رياضية. وهذه العناصر الأولية يتم ربطها من خلال بديهيات رياضية (Axioms) كما يتم تخزين التراكيب الجديدة فى مكتبة (Library). كما يمكن بناء تراكيب أكبر وضمها إلى المكتبة وبالتالى يمكن بناء مكتبة ضخمة من التراكيب المدققة رياضيا.

والشكل (٤٣ - ٢) يوضح الخصائص الرئيسية لأداة توصيف المتطلبات التى

تستخدم أدوات الـ (CASE).

A proper specification should :

* Be free from errors.
* Have conceptual clarity.
* Be easy to understand by manager, analysis, and programmers.
* Be presentable in varying degrees of detail.
* Be easy to create.
* Be computable (i.e. have enough precision that program code can be generated automatically).
* Be formal input to a program code generator.
* Be easy to change.
* Be complete.
* Be traceable when changes are introduced.
* Be independent of hadware.
* Employ a data dictionary.
* Employ a data model based on formal data analysis.
* Employ a program module library with automatic verification of interface.
* Employ computerized tools that make it easy to manipulate and change.

شكل (٤٣ - ٢)



الفصل الرابع والأربعون

منهجية (HOS)



٤٤ - ١ مقدمة

فى الطرق التقليدية يتم تحليل الوظائف الخاصة بالنظام بأى صورة يراها مصمم النظام مناسبة له. ولكن فى المنهجية التى سيتم دراستها فى هذا الفصل يتم تحليل وظائف النظام بطريقة رياضية محددة مما يجعلها مدققة رياضيا كما يوفر علاقات رياضية محددة بين الوظائف. ويستمر تحليل الوظائف حتى نصل إلى بلوكات يتم بواسطتها توليد الكود.

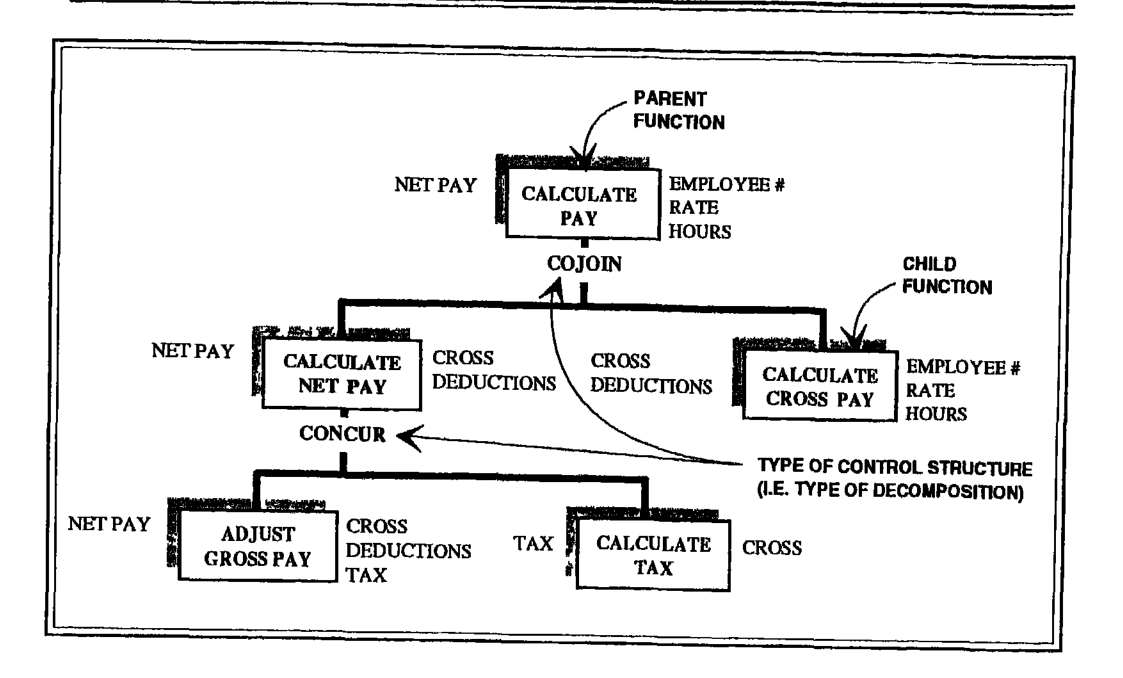
والمنهجية المشروحة فى هذا الفصل تسمى (HOS) وهى اختصار (Higher Order Software) وقد تم تطبيقها بواسطة أداة (CASE) تسمى (USE. IT). وهذا الأداة تولد الكود آليا لأعقد النظم.

٤٤ - ٢ الشجرة الثنائية (Binary Tree)

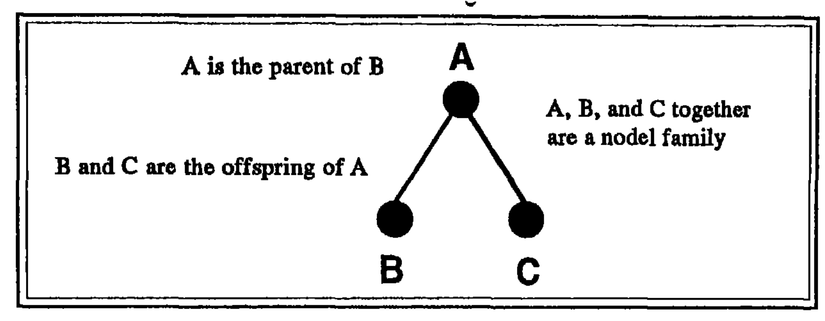
تستخدم منهجية (HOS) الشجرة الثنائية فى تحليل وظائف النظام ويتم من خلالها تمثيل الوظائف على هيئة عقد (Nodes) تبدأ من عقدة الجذر التى تتفرع إلى عقدتين تتفرعان إلى عقد أخرى وهكذا. ولكن الفرق بين تحليل الوظائف بهذه الطريقة وبين الشجرة الثنائية التقليدية أن كل عملية تحليل (Decomposition) تكون من نوع خاص يسمى تركيبة التحكم (Control Structure). وهناك أنواع مختلفة من تراكيب التحكم ويتم كتابة نوع تركيبة التحكم المستخدمة تحت العقدة الخاصة بها. أنظر شكل (٤٤ - ١).

وفى مخططات (HOS) التى تسمى أيضا خرائط التحكم (Control Maps) تكون كل عقدة فرعية أسفل العقدة الأب (Parent) وتسمى العقدة الفرعية فى هذه الحالة الفرع (Offspring). فمثلا شكل (٤٤ - ٢) يوضح عقدة الأب (Parent) والأفرع (Offsprings) الخاصة بها.

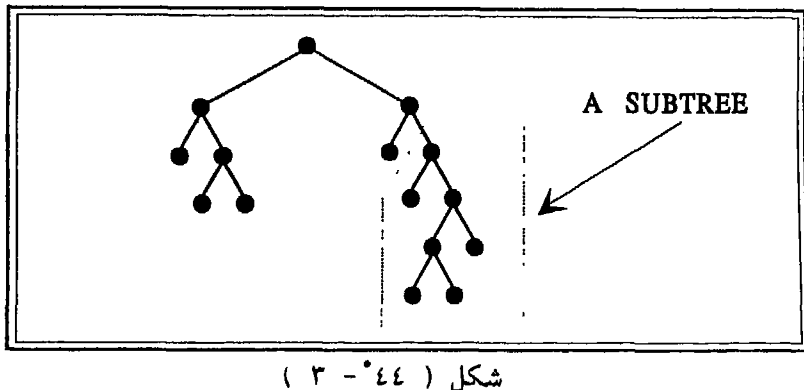
ويمكن إعتبار إحدى العقد فى الشجرة جذرا لشجرة أخرى مما يؤدى إلى وجود شجرة داخل شجرة وتسمى فى هذه الحالة شجرة فرعية. أنظر شكل (٤٤ - ٣)



شكل (٤٤ - ١)



شكل (٤٤ - ٢)

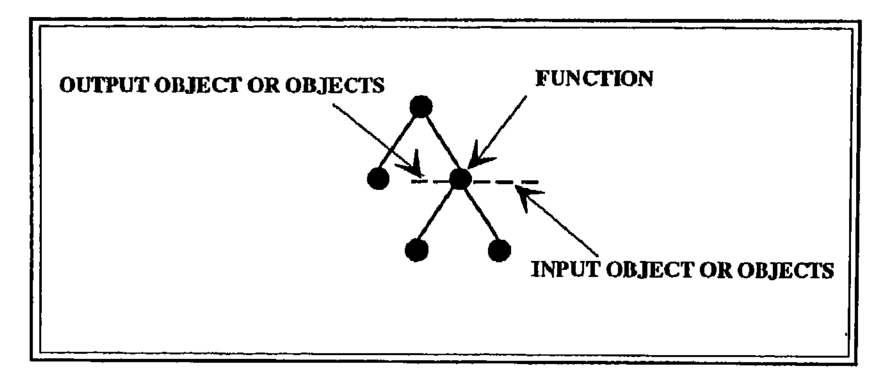


شكل (٤٤ - ٣)

٤٤ - ٣ الوظائـــف (Functions)

كل عقدة فى شجرة(HOS) تمثل وظيفة معينة. وكل وظيفة لها مدخلات (Input) تتمثل فى أشياء (Objects) معينة ولها كذلك مخرجات تتمثل فى أشياء أخرى. وهذه الأشياء قد تكون عنصر بيانات ، قائمة ، جدول ، تقرير ، ملف ، أو أشياء طبيعية مثل موظف أو طالب أو عربة ... الخ.

والمدخلات يتم كتابتها يمين العقدة الممثلة للوظيفة كما يتم كتابة المخرجات يسار العقدة وذلك للتمشى مع الإصطلاح الرياضى فى تعريف الوظيفة (Function) أنظر شكل (٤٤ - ٤)



شكل (٤٤ - ٤)

والوظيفة قد تكون رياضية (Mathematical) كالآتى مثلا

Y = Square Root of (X)

حيث تمثل (X) المدخلات وتمثل (Y) المخرجات وتمثل (Square Root of) الوظيفة.

وقد تكون الوظيفة أمرا من أوامر الحاسب كالآتى مثلا

$Stack_2$	ADD ITEM TO STACK	$Stack_1$ Integer

حيث تمثل (Stack1, Integer) المدخلات وتمثل (Stack2) المخرجات وتمثل (ADD ITEM TO STACK) الوظيفة.

٤٤ - ٤ من المتطلبات إلى التصميم التفصيلى

توضح مخططات (HOS) الشجرية كيف يتم تحليل الوظائف الكبيرة إلى وظائف فرعية حيث يمثل الجذر أكبر تعبير (Statement) يمثل الوظيفة. وعندما تصل الشجرة إلى الأوراق (Leaves) فإننا نصل إلى وظائف لاتحتاج إلى تحليل آخر. وهذه الأوراق (Leaves) إما أن تكون وظائف أولية (Primitive Functions) أو وظائف موجودة داخل مكتبات (Libraries) أو وظائف خارجية (External).

وفى منهجيات متعددة تكون المطالب (Requirement) والمواصفات (Specifications) والتصميم التركيبى والتصميم التفصيلى مكتوبين بلغات مختلفة وغالبا تكون غير متوافقة. أما فى منهجية (HOS) فإن هناك لغة واحدة مستخدمة فى جميع هذه المراحل. ويلاحظ دائما أن شجرة التحليل تتحرك من الوظائف الكبيرة أو الواسعة إلى التصميم التفصيلى. وفى نفس الوقت يتم إختبار الأخطاء والأجزاء الناقصة أو المحذوفة فى كل مرحلة من مراحل التحليل. وتستخدم الأوراق (Leaves) الموجودة فى الشجرة فى توليد الكود آليا.

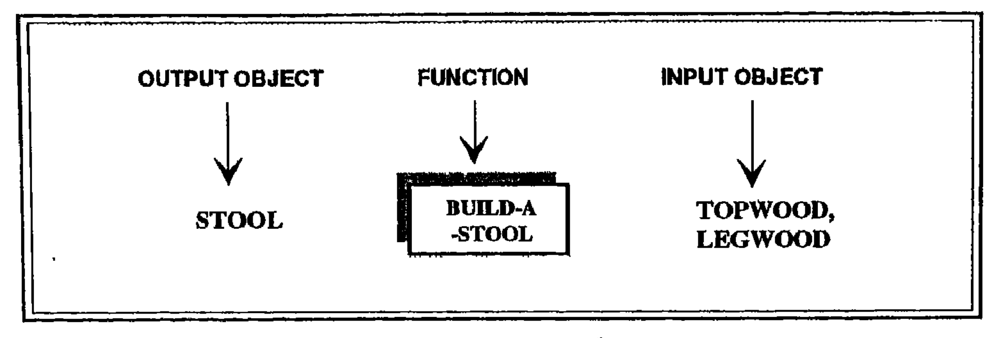
٤٤ - ٥ تراكيب التحكم الأولية

تتميز شجرة (HOS) بأن عملية تحليل الوظيفة إلى وظائف فرعية (أو بعبارة أخرى العلاقة بين الوظيفة وفرعيها) تكون مدققة رياضيا. ويتم تحليل الوظيفة إلى أحد الفرعين (Offsprings) باستخدام مايسمى بتركيبة التحكم (Control Structure). وهناك ثلاثة تراكيب تحكم أولية تستخدم فى الشجرة وهى : (JOIN) ، (INCLUDE) ، (OR). وهناك تراكيب تحكم أخرى يمكن الحصول عليها بالدمج بين هذه التراكيب الأولية.

٤٤ - ٥ - ١ التركيبـــة (JOIN)

نفرض أن هناك وظيفة مثل (BUILD A STOOL) وتعنى بناء كرسى بدون ظهر. ويفرض أن الكرسى يتم إنتاجه من نوعين من الخشب (TOPWOOD) و (LEGWOOD). أى أن هناك نوعا خاصا بسطح الكرسى (Top) وهناك نوعا

خاصا بالأرجل (Legs). والشكل التالى يوضح الوظيفة ومدخلاتها ومخرجاتها.



شكل (٤٤ - ٥)

وفى الرياضيات نكتب (Y = F(X)) حيث (Y) تمثل نتيجة تطبيق الوظيفة (F) على البيانات (X). وبالمثل يمكن تمثيل متطلبات إنشاء الكرسى كالآتى :

STOOL = BUILD-A-STOOL (TOPWOOD,LEGWOOD)

ولإنشاء الكرسى فإننا نحتاج إلى وظيفتين أو عمليتين أحدهما لإنشاء الأرجل (Legs) وسطح الكرسى (Top) والأخرى لتجميع أجزاء الكرسى. لذلك فإن الوظيفة (BUILD-A-STOOL) يمكن تحليليها إلى الوظيفتين الفرعيتين (MAKE-PARTS) و (ASSEMBLE-PARTS) والوظيفة(MAKE-PARTS) تخرج منها المخرجات (TOP) ، (LEG). هذه المخرجات تكون مدخلات للوظيفة (ASSEMBLE-PARTS).

ولتوضيح ذلك بالمعادلات فإن الوظيفة

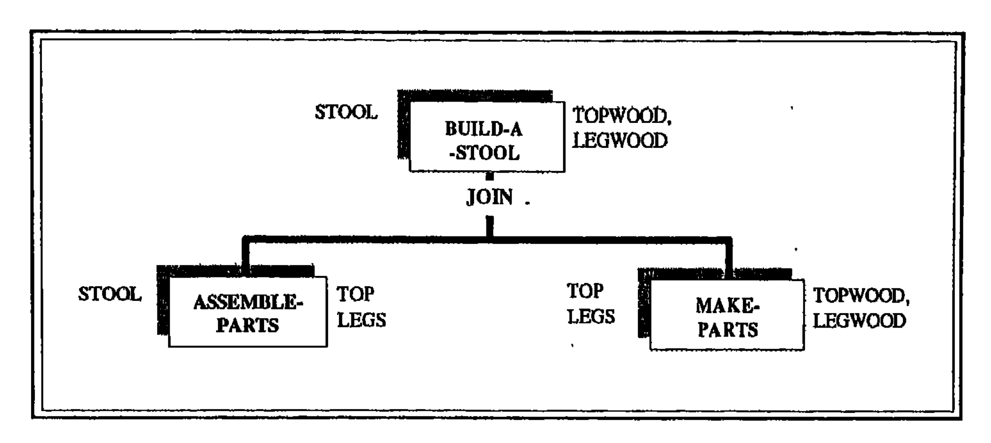
STOOL = BUILD-A-STOOL (TOPWOOD,LEGWOOD)

يتم تحليلها إلى الوظيفتين

TOP,LEGS = MAKE-PARTS(TOPWOOD,LEGWOOD)
STOOL = ASSEMBLE-PARTS(TOP,LEG)

ولتمثيل هذه الوظائف على هيئة شجرة (HOS) يتم رسم الشجرة الموضحة

بالشكل (٤٤ - ٦)



شكل (٤٤ - ٦)

ويوضح الشكل استخدام تركيبة التحكم الأولية (JOIN). ويلاحظ أن أحد الفرعين يعتمد على الآخر. وذلك لأن المخرجات الخاصة بالوظيفة اليمنى (TOP,LEGS) يجب أن تكون مدخلات للوظيفة اليسرى. ويلاحظ أن المدخلات للوظيفة اليمنى هى نفس مدخلات الوالد (Parent). كما أن المخرجات الخاصة بالوظيفة اليسرى هى نفس مخرجات الوالد. أى أن نفس التأثير الخاص بوظيفة الوالد يتم إعادة إنتاجه بواسطة الوظيفتين الفرعيتين. والمخطط يقرأ من اليمين إلى اليسار. أى أن (TOPWOOD) ، (LEGWOOD) يتم إدخالهما إلى الوظيفة (MAKE-PARTS) وينتج عن ذلك المخرجات (TOP) ، (LEGS) التى تصبح مدخلات للوظيفة (ASSEMBLE-PARTS) التى ينتج عنها الكرسى (STOOL).

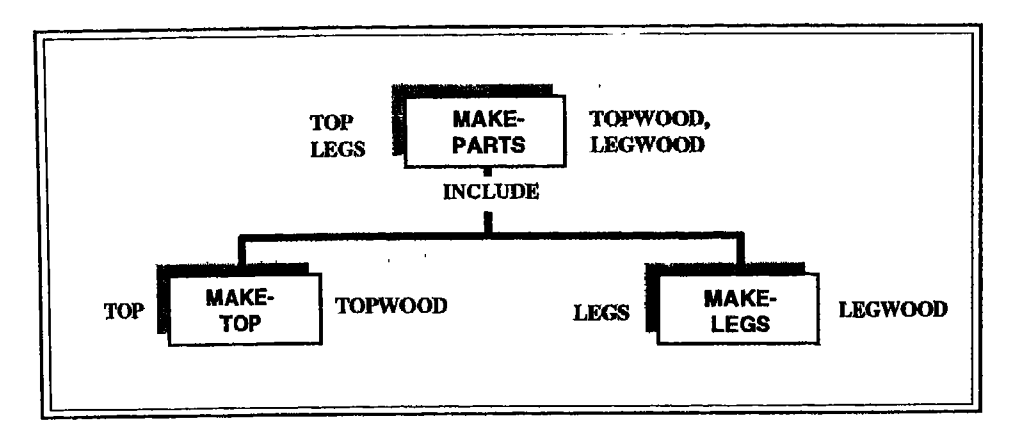
٤٤ - ٥ - ٢ التركيبة (INCLUDE)

يمكن تحليل الوظيفة(MAKE-PARTS) إلى وظيفتين وهما(MAKE-TOP)، (MAKE-LEGS). ويتم صناعة سطح الكرسى من الخشب (TOPWOOD) والأرجل من الخشب (LEGWOOD). أى أن الوظيفتين الجديدتين تكونان كالآتى :

TOP = MAKE-TOP(TOPWOOD)
LEGS = MAKE-LEGS(LEGWOOD)

ولتمثيل هذه الوظائف على هيئة شجرة (HOS) يتم رسم الشجرة الموضحة

بالشكل (٤٤ - ٧)



شكل (٤٤ - ٧)

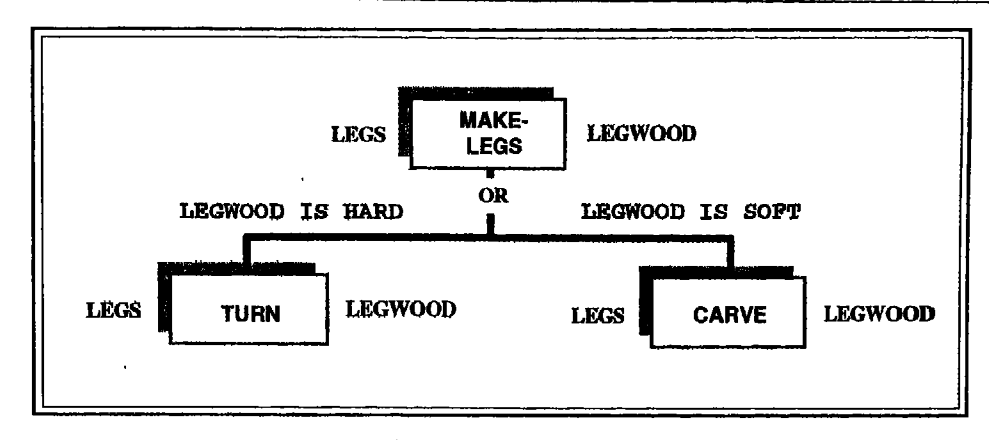
وفى هذه التركيبة يلاحظ أن الفرعين مستقلان تماما. أى أنه ليس هناك علاقة بين المدخلات أو المخرجات الخاصة بهما. وهذا يعنى أنه يمكن تشغيل كل منهما منفصلا عن الآخر. ويلاحظ أن كلا الفرعين يستخدم مدخلات الوالد (Parent) كما أن كليهما ينتجان مخرجات الوالد.

٤٤ - ٥ - ٣ التركيبة (OR)

نفرض أن الأرجل (LEGS) يمكن تصنيعها بإحدى طريقتين إما بعملية تشكيل (TURN) أو بعملية نحت (CARVE). لذلك فإننا نحصل على الأرجل (Legs) من إحدى الوظيفتين التاليتين :

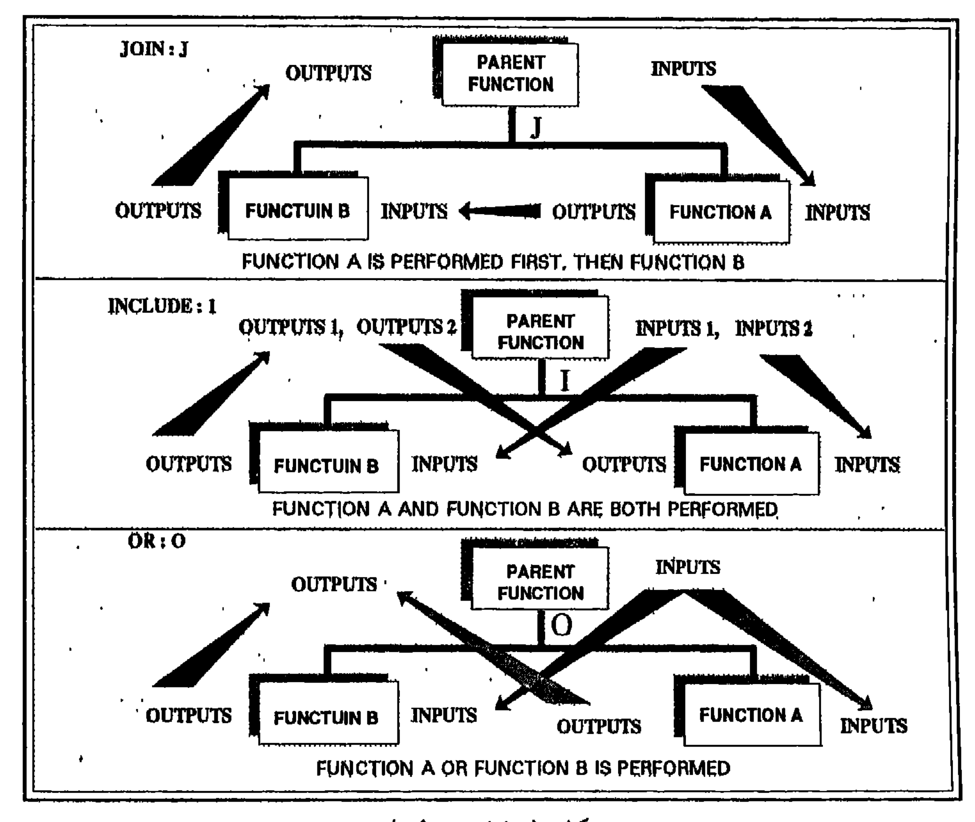
LEGS = TURN(LEGWOOD)
LEGS = CARVE(LEGWOOD)

ولتقرير أى العمليتين يتم اختيارها يستخدم تعبير منطقى تكون نتيجته إما صحيح (True) أو غير صحيح (False). والشكل (٤٤ - ٨) يوضح شجرة (HOS) المستخدمة فى هذه الحالة



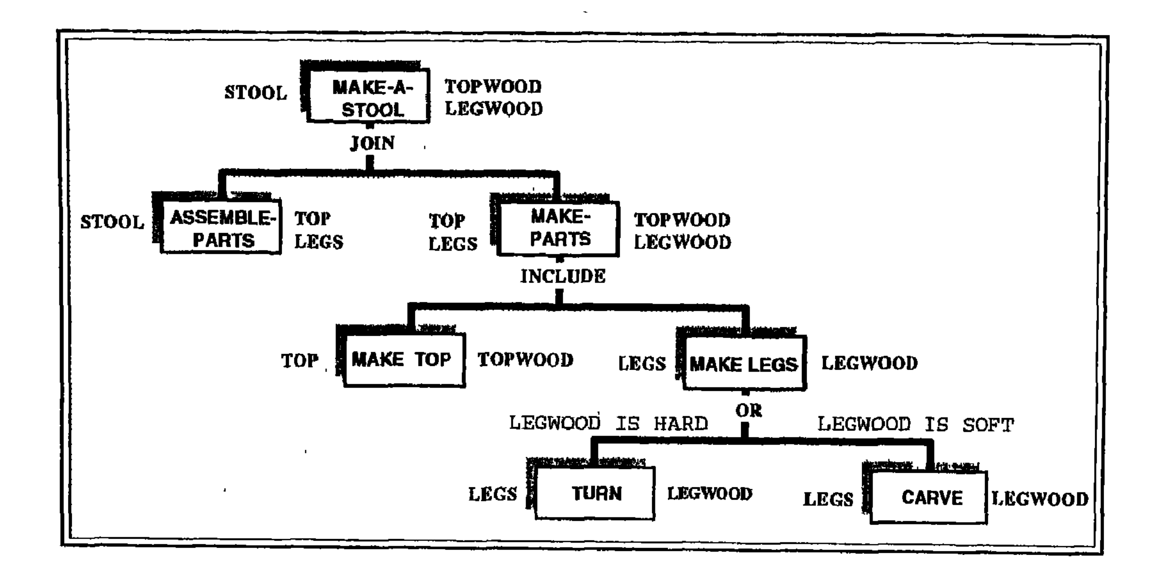
شكل (٤٤ - ٨)

وفى هذه التركيبة يلاحظ أن أحد الفرعين يستقبل التكليف من الوالد وليس الإثنان. ويلاحظ أيضا أن المخرجات الناتجة من الفرعين هى نفس مخرجات الوالد وهى(LEGS). والشكل (٤٤ - ٩) يلخص تراكيب التحكم الأولية السابق شرحها.



شكل (٤٤ - ٩)

ويمكن دمج الوظائف السابقة فى شجرة واحدة كالموضحة بالشكل (٤٤ - ١٠) وتسمى الشجرة فى هذه الحالة خريطة التحكم (Control Map).



شكل (٤٤ - ١٠)

٤٤ - ٦ توليـــد الكـــود

عن طريق تحليل الوظائف ثنائيا بواسطة تراكيب التحكم(JOIN)،(INCLUDE)، (OR) فإننا نستطيع الحصول على تراكيب تحكم يمكن تدقيقها رياضيا. ومع إستمرار عمليات التحليل (Decomposition) فإننا نصل إلى عقد الأوراق (Leaf Nodes). وهذه الأوراق إما أن تكون عقد أولية (Primitives) أو فروع (Subroutines) مبنية أيضا من خلال منهجية (HOS). ويصبح التصميم كاملا عندما تصل كل الفروع إلى هذه الأوراق ويمكن فى هذه الحالة توليد الكود آليا من هذه الأوراق. وكل مرحلة من مراحل التصميم يمكن اختبار صحتها وتدقيقها آليا.

وهناك أربعة أنواع من أوارق العقد (Leaf Nodes) تتلخص فى الآتى :

١ - عمليات أو وظائف أولية (Primitive Operations) ويرمز لها بالرمز (P). وهى الوظائف التى لايمكن تحليلها إلى وظائف أدنى. وهى تكون محددة بدقة

ويمكن تدقيقها رياضيا.

٢ - عمليات معرفة فى أماكن أخرى (Operations Defined Elsewhere) ويرمز لها بالرمز (OP). وهى وظائف سوف يتم تحليلها فى خريطة تحكم موجودة إما فى نفس التصميم أو فى مكتبة (Library).

٣ - عمليات إستدعاء ذاتى (Recursive Operations) وهى عقد خاصة تتيح بناء الحلقات التكرارية (Loops).

٤ - عمليات خارجية (External Operations) وهى عبارة عن برامج خارجية ليست مكتوبة بواسطة منهجية (HOS). وهذه البرامج لايمكن تدقيقها أو ضمان صحتها.

٤٤ - ٧ الأداة (USE. IT)

لاشك أن تطبيق منهجية (HOS) يدويا يعتبر عملية معقدة ومملة بالنسبة لمعظم محللى النظم. لذلك فإن استخدامها يكون مرتبطا بأداة أخرى تسمى (USE. IT) وهذه الأداة توفر الآتى :

١ - لغة للتعبير عن الوظائف وتحليلها إلى وظائف أدنى.

٢ - شاشات متفاعلة تتيح للمستخدم بناء وتصحيح خرائط التحكم (Control Mpas).

٣ - مكتبة تتضمن أنواع البيانات (Data Types) والوظائف الأولية (Primitive Functions) والروتينات المعرفة سابقا.

٤ - روتين خاص بالتحليل (Analyzer) وظيفته التأكد من أن كل القواعد التى تعطى منطق صحيح (Correct Logic) قد تم تطبيقها.

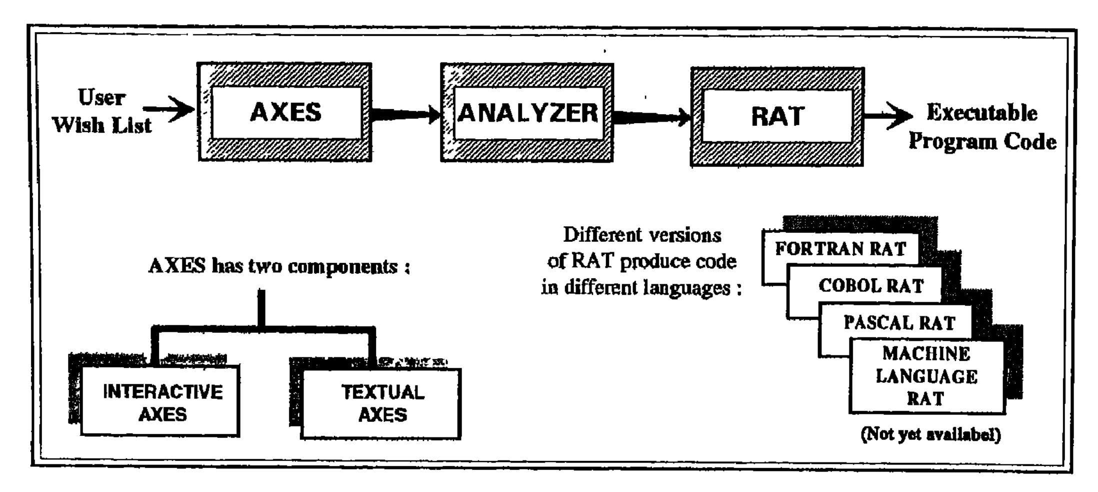
٥ - مولد وظيفته توليد الكود آليا.

وتتكون الأداة (USE. IT) من ثلاثة أجزاء رئيسية وهى (AXES) ، (ANALYZER) ، (RAT) كما يتضح من الشكل (٤٤ - ١١)

٤٤ - ٨ كيف تعمل الأداة (USE. IT) ؟

كما سبق الإيضاح فإن تحليل الوظائف ثنائيا بواسطة التراكيب (JOIN) ، (INCLUDE) ، (OR) يؤدى إلى الوصول إلى تراكيب تحكم يمكن تدقيقها رياضيا. ومع استمرار التحليل فإننا نصل إلى عقد الأوراق (Leaf Nodes). وعقد الأوراق هى إما

عقد أولية (Primitives) مدققة رياضيا أو روتينات فرعية مبنية بواسطة (HOS). وعندما تصل كل الفروع إلى عقد الأوراق فإن التصميم يصبح كاملا كما يمكن توليد الكود آليا من هذه العقد.



شكل (٤٤ - ١١)

والجزء (AXES) من البرنامج هو المسئول عن توصيف الوظائف وأنواع البيانات وتراكيب التحكم. وهو يختلف من لغة البرمجة فى أنه يستطيع التعبير عن متطلبات واسعة كالآتى مثلا

STOOL = MAKE-STOOL(TOP,LEGS,LATHE,HANDTOOLS)

وهذه التعبيرات الواسعة يتم تحليلها حتى يتم الوصول إلى التراكيب الأولية التى يتم بواسطتها توليد الكود.

ويتكون الجزء (AXES) من جزئين رئيسين. الجزء الأول هو برنامج رسم قوى يتيح للمستخدم تصميم شجرة (HOS) والتعامل معها من خلال شاشات تفاعلية واضحة. والجزء الثانى هو برنامج يحول المعلومات التى يتم انتاجها بواسطة الجزء الأول إلى نسخة نصية (Textual) تتيح للمحلل (ANALYZER) تحليل هذه المعلومات.

ويقوم المحلل (ANALYZER) بالتأكد من تطبيق القواعد الرياضية والمنطقية. وهو يختبر أخطاء القواعد أولا ثم يختبر عمليات نقل البيانات والتأكد أنه لم يتم حذف أو تغيير بيانات المتطلبات.

أمـا الـ (RAT) وهـو اختصـار (Resourse Allocation Tool) فيستخدم مخرجات المحلل(ANALYZER) ليولد الكود آليا. وهناك نسخ مختلفة من الـ (RAT) تولد كود البرنامج بلغات مختلفة مثل(FORTRAN RAT) ، (PASCAL RAT) ، (COBOL RAT) وهكذا. وأفضـل هذه النسـخ هـى النسخـة التـى تـولـد كـود الآلـة (Machine Code) لأنه لايحتاج إلى ترجمة (Compilation).

الملاحق



ملحق (١)
مجموعة كتب دلتا



- David Kroenke, colorado State University." Database Processing "

- Daniel F. Stubbs and Neil W. Webre, California Polytechnic State University, Sanluis Obisco " Data structures " ,1985

- Seymour Lipschutz, Mcgraw - Hill Book Company, " Data Structures " 1986

- Alan Simpson, " Understanding Dbase 111+ ", 1986 C . J DATE , Addison - Wesley . Publishing Company, " Database Systems ", 1986

- Shari Lawrence Pfleeger, Macmillan Publishing Company, New York, " Software Engineering ", 1991

- James Martin , Carina Mcclure, Printice Hall , Englewood Cliffs, New Jersey, " Structured Techniques : The basis for CASE " , 1988



ملحق (٢)

قائمة المراجع



١ الحاسبات الإلكترونية حاضرها ومستقبلها

يعتبر هذا الكتاب من أهم الكتب التى يحتاج القارىء اليها سواء كان فى بداية طريق دراسة علوم الحاسب أو قطع شوطا كبيرا فى هذا المجال . ذلك لأن هذا الكتاب يتضمن معلومات عن كل مايتعلق بتكنولوجيا الحاسب بدءا من إستعراض تطور الحاسبات من حيث المكونات المادية والبرامج وتطور نظم التشغيل وانتهاء بلغات الجيل الرابع ونظم دعم القرار مرورا بجميع الموضوعات التى تشغل المتخصصين فى مجال الحاسب مثل تعريب الحاسبات ولغات الحاسب بالإضافة إلى البرامج التطبيقية المختلفة مثل نظم إدارة قواعد البيانات والجداول الإلكترونية وبرامج تنسيق الكلمات ونظم إدارة المشروعات ونظم التصميم الهندسى هذا بالإضافة إلى موضوعات أمن البيانات وفيروسات الحاسب . ويحتوى هذا الكتاب على جزء خاص بمستقبل تكنولوجيا الحاسبات يتضمن الذكاء الإصطناعى والنظم الخبيرة والبرمجة الشيئية (Object Oriented) والمعالجة العصبية للمعلومات (Neural Networks) ومعالجة اللغات الطبيعية واللغة العربية بالحاسب والكثير من الموضوعات الأخرى المرتبطة بهذا المجال . والكتاب يحتوى على مالايقل عن ٨٥٠ صفحة مزودة بأكثر من ٥٠٠ شكل توضيحى .

٢ دائرة معارف الحاسب الإلكترونى

يعتبر هذا الكتاب من المراجع العلمية المتميزة فى مجال تكنولوجيا المعلومات. فهو إلى جانب مايتمتع به من دقة وشمول فإن أسلوبه يتميز بالسهولة والوضوح دون الإخلال بالمضمون العلمي. والكتاب لايقتصر على الترجمة الدقيقة لمصطلحات الحاسب ، وإنما يوفر أيضا الشرح التفصيلى لهذه المصطلحات وأى معلومات مرتبطة بها . وقد روعى عند إعداد هذه الموسوعة أن يجد فيها القارئ كل غايته ، بدءا من القارئ العادى الذى يسعى إلى الحصول على المعلومات البسيطة الشاملة ، وانتهاء بالقارئ الفنى والمتخصص الذى يسعى إلى الحصول على

معلومات فنية دقيقة. لذلك فقد تم تغذية الموسوعة بآخر ماوصل إليه العلم فى مجال تكنولوجيا المعلومات لملاحقة التطور السريع فى هذا المجال. ويحتوى الكتاب على مالايقل عن ألف ومائتى مصطلح مرتبة بالترتيب الهجائى للحروف حتى يستطيع القارئ بسهولة الوصول إلى المصطلح المطلوب. ويصل عدد صفحات الكتاب إلى ٥٠٠ صفحة مزودة بما يزيد عن ٣٠٠ شكل توضيحى

٣ المرجع الشامل لنظام التشغيل (DOS)

يعتبر هذا الكتاب من المراجع العربية المتميزة التى تتناول نظام التشغيل (DOS) وتوضح خصائصه الفنية وأوامره ووظائفه بشرح يتصف بالبساطة إلى جانب الدقة والشمول . والكتاب لايقتصر على نظام التشغيل (DOS) فقط ، ولكنه يتناول أيضا نظام التشغيل (DOS-4)، (DOS-5) ، (DRDOS-6) بالإضافة إلى نظام النوافذ (Windows) الذى يوفر التفاعل الجيد بين المستخدم والحاسب . كما يتناول الكتاب أيضا أهم الأدوات المساعدة لنظام التشغيل (DOS) مثل برنامج (PC Tools) وبرنامج(Norton) . وهى الأدوات التى تساعد المستخدم على إستعادة الملفات الممسوحة بطريق الخطأ وكذلك فحص القرص واكتشاف أعطاله وإصلاحها وتحسين أداء القرص وتحسين أداء الحاسب بصفة عامة . كما يتناول الكتاب أيضا أهم السلبيات والمشاكل التى يمكن أن يتعرض لها نظام التشغيل (DOS) ممثلة فى فيروسات الحاسب مع توضيح أخطار هذه الفيروسات وطرق التغلب عليها والوقاية منها . والكتاب يتكون من ستة أجزاء بالإضافة إلى الملاحق ، و يزيد عدد صفحاته عن ٦٥٠ صفحة محتوية على مايزيد عن ٦٠٠ شكل توضيحى .

٤ عالم الجداول الإلكترونية (بين الدراسة والتطبيق)

يعتبر هذا الكتاب من أهم الكتب التى تناولت برامج الجداول الإلكترونية بالشرح التفصيلى الدقيق مع الأسلوب السهل الواضح . ورغم أنه يشرح ثلاثة من البرامج تمثل أقوى

برامج الجداول الإلكترونية على الإطلاق وهى برامج :

LOTUS 123 - EXCEL - QUATRO PRO

إلا أنه يتضمن أيضا شرحا وافيا لأساسيات التعامل مع برامج الجداول الإلكترونية بصفة عامه مما يساعد المستخدم على الإلمام بأساليب التعامل مع جميع برامج الجداول الإلكترونية . ويوفر البرنامج شرحا لأهم الخصائص الفنية المتقدمة مثل استخدام الماكرو واستخدام خصائص قواعد البيانات واستخدام النوافذ وربط الجداول الإلكترونية واستخدام مكتبات الربط وكذلك استخدام الأنواع المتقدمة من الرسومات مثل الرسومات ثلاثية الأبعاد واستخدام الشاشات المنزلقة . كما يوفر الكتاب شرحا تفصيليا لطريقة حل مسائل البرمجة الخطية عن طريق الجدول الإلكترونى مع توضيح ذلك بمثال عملى واضح . كما يشرح الكتاب تطبيقا شاملا على الجداول الالكترونية وهو بعنوان "إدارة التدفق النقدى" وذلك فى اكثر من ٢٥٠ صفحة متضمنة عددا كبيرا من الرسوم التوضيحية والمخططات . وهذا التطبيق يساعد مدير العمل على متابعة التدفق النقدى فى منشأته والسيطرة عليه . والكتاب فى مجمله يزيد عدد صفحاته عن ٦٧٠ صفحة متضمنة مالا يقل عن ٦٠٠ شكل توضيحى .

٥ الحاسب الإلكترونى وقواعد البيانات (الجزء الأول)

يعتبر هذا الكتاب من أهم الكتب التى تتناول قواعد البيانات بصفة عامة وبرامج عائلة (DBase) بصفة خاصة وهى البرامج :

DBASEIII+ - DBASEIV - FOXBASE+ - FOXPRO

والكتاب يوضح مفهوم قواعد البيانات ومفهوم إدارة قواعد البيانات . كما يشرح الكتاب بالتفصيل أهم الجوانب الفنية المرتبطة ببرامج عائلة (DBase) متضمنة قواعد إنشاء هيكل الملف (DBase Structure) وقواعد تصميم شاشة الإدخال وعرض السجلات على الشاشة وتصحيحها وقواعد تنظيم ملف قاعدة البيانات عن طريق الفرز (Sorting) والفهرسة

(Indexing) وطـــرق البحث عـــن الســجلات واستخدام ملفات البحث (Query Files) وطباعة التقــارير وربط قــواعد البيانات. كما يشرح الكـتـاب اســتـخدام أوامر النقطة (Dot Com-mands) وقواعد كـتابـة البرامــج الخــاصــة بقواعد البيانات وإستخدام متــغيرات الذاكـــرة (Memory Variables) وملفات الذاكرة (Memory Files) وملفات الخطوات (Procedure Files) والدوال المستخدمة وطرق التحكم فى شاشة الإدخـــال وكذلك التحكم فــى الطباعة ووسائل تصحيح الأخطاء (Debugging Tools) . ويتكــون الكتــاب مـن ستة وعشرين فصلا بالإضافة إلى أربعة ملاحق.كما يحتوى على العديد من الأشكال التوضيــحية ويزيد عدد صفحاته عن ٣٨٠ صفحة.

٦ الحاسب الإلكترونى وقواعــــد البيانات (الجزء الثانى)

يعتبرهذا الكتاب جزءا مكملا للجزء الأول ويحتـوى على شـرح تفصيلى للأوامـر والدوال المسـتخدمـة فى برامج عـائلة (DBase) و يكون مـع الجــزء الأول المرجع الشــامل الذى يعين المستخدم على كتابة البرامج التطبيقية عالية الكفاءة والتى تخدم جميع مجالات نظم المعلومات . ويزيـــد عــــدد صفحـــات الكتــــاب عن ٢٤٠ صفحــة متضمنـة العديـــد مــن الأشكال التوضيحية .

٧ تطبيقات نظم إدارة قواعـــــد البيانــــات

يعد هذا الكتاب إضافة حقيقية للمكتبة العربية التى تفتقر إلى هذا النوع من الكتب التى تتناول تطبيقات عملية لنظم إدارة قواعد البيانات . ولا يكتفى الكتاب بالشرح الإجمالى لكل نظام والبرامج المكونة له ، ولكنه يقف عند كل سطر فى البرامج ويشرحه شرحا دقيقا موضحا البدائل المختلفة ومميزات وعيوب كل من هذه البدائل . والكتاب يتكون من ستة أجزاء . الجـزء الأول يحتوى على مراجعة للكتاب الأول "نظم إدارة قواعد البيانات" بجزأيه الأول والثانى . والجزء الثانى من الكتاب يشرح نظام معلومات شئون الطلبة الذى يصلح للاستـخدام فى أى

مؤسسة تعليمية لمتابعة بيانات الطلبة والسيطرة الكاملة على إدخال البيانات وعرضها وتصحيحها وطباعة التقارير . والجزء الثالث من الكتاب يشرح نظام المخازن كنموذج لقواعد البيانات التى تتعامل مع ملفات الحركة (Transaction Files) . والجزء الرابع يشرح نظام حسابات العملاء كنموذج للبرامج التى تستخدم ملفات الخطوات (Procedure Files) لتقليل عدد الملفات المفتوحة . والجزء الخامس يشرح بعض الأدوات والوسائل المتقدمة فى كتابة البرامج من خلال ثلاثة برامج مختلفة أحدها يستخدم فى كتابة الشيكات ، والثانى يتيح للمستخدم إختيار الألوان التى يفضلها فى شاشات إدخال البيانات ، والثالث يتيح عرض شاشات إدخال بيانات تحتوى على عمود ضوئى يمكن تحريكه الى الإختيار المطلوب. والجزء السادس يشرح بعض التطبيقات الإضافية ويتضمن أيضا شرح مواد التطبيقات الخاص ببرنامج (DBase III+). والكتاب يزيد عدد صفحاته عن خمسمائة صفحة متضمنة مايزيد عن ١٠٠ شكلا توضيحيا .

٨ فيروسات الحاسب وأمن البيانات

يتناول هذا الكتاب قضية أمن البيانات بصفة عامة موضحا الأساليب التكنولوجية المختلفة لتنفيذ ذلك مثل استخدام التشفير وإعادة التشفير واستخدام كلمات السر تبعا لمستويات السرية المختلفة وارتباط ذلك بنظم التشغيل. ثم يشرح الكتاب موضوع فيروسات الحاسب باعتباره من أهم الموضوعات التى تشغل عقول كثير من المهتمين بمجال الحاسب نظرا لما يمثله الفيروس من خطورة على أمن البيانات . ويقدم الكتاب دراسة موضوعية دقيقة تتناول التحليل الدقيق للفيروس من حيث تكوينه وخصائصه الفنية بما يتيح للمستخدم التعرف السليم على هذا الموضوع بعيدا عن التخيلات والأوهام . كما يشمل الكتاب أيضا توضيحا لطرق الوقاية والعلاج والأمصال البرمجية المستخدمة ضد أنواع معينة من الفيروسات . ويتضمن الكتاب بعض الملاحق يناقش أحدها أشهر نماذج فيروسات الحاسب مع شرح دقيق لمواصفات أكثر من ١٥٠ فيروس من فيروسات الحاسب .

٩ الحاسب ونظم المعلومات الإدارية

يتناول هذا الكتاب أهمية إستخدام الحاسب فى نظم المعلومات الإدارية وأهم الموضوعات المرتبطة بتكنولوجيا المعلومات وتطور الحاسبات ونظم التشغيل ولغات الحاسب . كما يوضح أساسيات تحليل وتصميم النظم بدءا من توصيف المتطلبات وتحليلها ثم تحليل بدائل تصميم النظام وكذلك استخدام أدوات هندسة البرامج (CASE Tools) . كما يتناول أهم تطبيقات الحاسب المالية والمحاسبية مثل نظام السيطرة على المخزون ونظم حسابات العملاء والحسابات العامة والمرتبات وإدارة التدفق النقدى والتسويق والتصنيع . كما يتناول الكتاب استخدام التقنيات الحديثة فى مجالات الأعمال مثل ميكنة المكاتب والإتصالات وأمن البيانات ونظم المعاونة فى اتخاذ القرار والذكاء الإصطناعى . والكتاب يزيد عدد صفحاته عن ٥٥٠ صفحة تتضمن مايزيد عن ٤٠٠ شكل توضيحى .

١٠ الحاسب الإلكترونى والذكاء الإصطناعى

يتناول هذا الكتاب تقنية من أحدث التقنيات التى ظهرت فى عصر الحاسب ، وهى التقنية الخاصة بالذكاء الإصطناعى مع تناول أحد المجالات التطبيقية الهامة المرتبطة بها بالتفصيل وهى النظم الخبيرة والتى بدأت تنتشر بسرعة كبيرة فى معظم أوجه الحياة العملية . وقد وضع فى الإعتبار أن يجد كل من القارئ المتخصص وغير المتخصص غايته من هذا الكتاب بحيث يتمكن القارئ من التفاعل بسلاسة وسرعة مع تكنولوجيا المستقبل . والكتاب يزيد عدد صفحاته عن ٣٥٠ صفحة متضمنة العديد من الأشكال التوضيحية .



مجموعة كتب "دلتا" لتكنولوجيا وعلوم الحاسب

١ - الحاسبات الالكترونية حاضرها ومستقبلها

٢ - دائرة معارف الحاسب الإلكتروني

٣ - المرجع الشامل لنظام التشغيل (DOS)

MS DOS 4 - MS DOS 5 - DR DOS 6

MS WINDOWS PCTOOLS

NORTON UTILITIES VIRUS-SCAN

٤ - عالم الجداول الالكترونية

(بين الدراسة والتطبيق)

LOTUS 123 - EXCEL - QUATRO PRO

٥ - الحاسب الإلكتروني وقواعد البيانات (الجزء الاول)

FOXBASE+ - DBASEIII+ - FOXPRO - DBASE IV

٦- الحاسب الإلكتروني وقواعد البيانات (الجزء الثاني)

٧ - تطبيقات نظم إدارة قواعد البيانات

٨ - فيروسات الحاسب وأمن البيانات

٩ - الحاسب ونظم المعلومات الإدارية

١٠ - الحاسب الإلكتروني والذكاء الإصطناعي

تعتبر المكتبة العربية ومحتوياتها في مجال التكنولوجيا من أكبر الدعائم الاساسية للمعرفة والتي تشكل بدورها أحد العوامل الرئيسية لجوانب التنمية المختلفة في المنطقة العربية . ولما كانت تكنولوجيا الحاسبات من أهم اتجاهات المعرفة التكنولوجية في الآونة الأخيرة فان قيمة المؤلفات تزداد في هذا الجانب من واقع ازدياد حاجة المستخدم العربي اليها . ومما لاشك فيه أن المكتبة العربية في مجال تكنولوجيا وعلوم الحاسب تعتبر فقيرة في هذا النوع من المؤلفات إلى درجة بعيدة نظرا لعدة جوانب نذكر منها مايلي :

- التسبق الفني اللازم والمواكب للتطور التكنولوجي السريع .
- افتقار المكتبة العربية الى القدر المطلوب من البعد العلمي اللازم للبعد الفني .
- الترابط الكامل بين جوانب المعرفة في المراجع المختلفة وعلاقات ذلك بدرجة استفادة القارئ وانعكاسه على درجة المعرفة ومستوى الخبرة .
- درجة ارتباطها بالتطبيق ومستوى استفادة القارئ منها .
- التغطية الكاملة لكل مستويات القراء مع اختلاف ثقافاتهم وخبراتهم .
- حاجة القارئ العربي في هذه المرحلة لتجاوز مستوى العديد من المراجع المتاحة والتي تعتمد على الترجمة الحرفية لدليل التشغيل للنظم التكنولوجية المختلفة الخاصة بالحاسب .

ومن هذا المنطلق فقد قامت مؤسسة دلتا باعداد مجموعة كتب " دلتا " لتكنولوجيا وعلوم الحاسب - والتي تتكون من العديد من المراجع - على أيدى نخبة مختارة من أساتذة الجامعات وكبار الخبراء المتخصصين في هذا المجال .

ومع التطور السريع في عالم تكنولوجيا الحاسبات وتعدد جوانب المعرفة المطلوبة للقارئ العربي فان موسوعة دلتا قد تم اعدادها على اساس التغطية الشاملة لاتجاهات التكنولوجيا الحديثة تبعا للأولويات المطروحة مع التغطية المستمرة للمستجدات في هذا المجال من خلال الاصدارات المختلفة لكتب الموسوعة على ضوء التطور السريع في مجال تكنولوجيا الحاسبات .

مجموعة كتب دلتا هي المرجع الشامل للدارسين والمتخصصين في مجال تكنولوجيا وعلوم الحاسب

دلتا كمبيوتر
Delta Computer
٥ عبد الله بن الزبير مصر الجديدة - نزهة - القاهرة - ت ٢٤٦٧٣٣٨ / ٢٤٤٠٣٧٥ / ٢٤٤٤٩٩٧ فاكسميلي ٢٤٣٥٦٦٥
5 ABDULLAH EBN EL-ZUBAIR NOZHA HELIOPOLIS, CAIRO Fax 2435665 Tel 2467338/2440375/2444997

مؤسسة " دلتا "

تعتبر مؤسسة " دلتا " من المكاتب الاستشارية الرائدة ذات الخبرات الفنية والعلمية الرفيعة والامكانات المتكاملة والمتميزة بتعدد التخصصات والخبرات فى نظم المعلومات الآلية .

وتتكون المؤسسة من عدد كبير من المتخصصين ذوى الخبرات الواسعة والعلميين من أساتذة الجامعات الممارسين للعديد من الحقول الفنية والبحثية المرتبطة بمجالات نظم المعلومات والاتجاهات المتطورة لميكنتها . فقد إتخذت المؤسسة الأساليب العلمية منهاجا فى تقديم الحلول للمشاكل المتنوعة والمعقدة والتى طالما تواجه العديد من المشروعات .

وتعتنى الدراسات الفنية التى يقوم بها خبراء المؤسسة بالعمل على تطوير الوسائل المناسبة للاستفادة من التكنولوجيا الحديثة فى مجالات نظم المعلومات . ويجدر الاشارة هنا بأن خبراءنا يشغلون العديد من المناصب القيادية ويقدمون الاستشارات العلمية والفنية للعديد من الهيئات والمؤسسات وحيث تجاوز مجال أنشطتهم الحدود المصرية الى المنطقة العربية كما يشغل بعض أعضاء المؤسسة مراكز أساسية فى اللجان الفنية الوطنية والعالمية فى الأعمال التى تتعلق بتخصصاتهم .

وعلى مدى أكثر من عشر سنوات قام خبراء ومستشارو مؤسسة " دلتا " بتحقيق العديد من الانجازات التى يمكن عرض بعض اتجاهاتها فيما يلى :

١ - القيام بدراسات الجدوى لادخال نظم الحاسبات الآلية فى الهيئات والمؤسسات المختلفة .
٢ - تحليل وتصميم وتنفيذ العديد من النظم الآلية والاشراف على المشروعات .
٣ - تصميم وتنفيذ البرامج التطبيقية للحاسبات الآلية فى العديد من مجالات نظم المعلومات والشئون المالية والادارية .
٤ - عمل الدراسات الخاصة بتقييم مستويات الأداء للنظم الآلية مع تحديد أساليب تطويرها .
٥ - تنفيذ برامج التدريب المتطورة على النظم الآلية المتخصصة .
٦ - القيام بالعديد من الأبحاث العلمية التى تتناول تعريب الحاسبات والقيام بالانجازات التطبيقية فى هذا المجال .

وأخيرا وليس آخرا فان مؤسسة " دلتا " قد أخذت على عاتقها مهمة اصدار سلسلة المراجع المتخصصة فى مجال تكنولوجيا وعلوم الحاسب حتى يستفيد منها أكبر عدد من القراء المتخصصين بالاضافة الى العديد من الدارسين فى مصر والعالم العربى .

والله الموفق ....



هذا الكتاب

الجزء الأول

مكونات قواعد البيانات

١ - مفاهيم أساسية
٢ - تمثيل البيانات
٣ - النموذج البياني
٤ - التصميم الهيكلي
٥ - تصميم قاعدة البيانات
٦ - قواعد البيانات العلائقية
٧ - لغة (SQL)
٨ - قواعد البيانات الشبكية
٩ - نظام (CODASYL) كنموذج لقواعد البيانات الهرمية
١٠ - قواعد البيانات الشبكية "النظام (ICMS)"
١١ - نظم إدارة قواعد البيانات

الجزء الثاني

نموذج شامل لنظم إدارة قواعد البيانات
(FoxPro, DBaseIV, DBaseIII+)

١١ - مقدمة
١٣ - إنشاء ملف قاعدة البيانات
١٤ - إنشاء شاشات الإدخال
١٥ - تعديل السجلات
١٦ - تنظيم
١٧ - البحث
١٨ - ملفات البحث
١٩ - التقارير والعناوين البريدية
٢٠ - ربط قواعد البيانات
٢١ - أوامر النقطة
٢٢ - كتابة البرامج
٢٣ - خصائص كتابة البرامج
٢٤ - تركيب البرنامج
٢٥ - متغيرات الذاكرة
٢٦ - أوامر التجهيز في البرنامج الرئيسي
٢٧ - التحكم في الشاشة من خلال البرنامج
٢٨ - التحكم في شكل ومدى المدخلات
٢٩ - الدوال المستخدمة مع المدخلات
٣٠ - مزيد من التحكم في شاشة الإدخال
٣١ - اختبار مدخلات المستخدم
٣٢ - التعامل مع قاعدة البيانات
٣٣ - التعامل مع البيانات
٣٤ - البيانات
٣٥ - التعامل مع بيئة الحاسب
٣٦ - استخدام وسائل أكثر تقدما
٣٧ - اختبار وتصحيح البرامج

الجزء الثالث

أوامر ودوال برامج (DBase)
(FoxPro, DBaseIV, DBaseIII+)

٣٨ - أهم الأوامر المستخدمة
٣٩ - أهم الدوال المستخدمة

الجزء الرابع

التحليل والتصميم وأدوات الـ(CASE)

٤٠ - مقدمة
٤١ - تحليل متطلبات النظام
٤٢ - تصميم النظام
٤٣ - نحو ميكنة أكبر لتطوير النظم
٤٤ - منهجية (HOS)



٥ عبد الله بن الزبير مصر الجديدة - نزهة - القاهرة - ت ٢٤١٧٣٣٨ / ٢٤٤٠٣٧٥ / ٢٤٤٤٩٩٧ فاكسيملي ٢٤٥٦٦٩٥
5 ABDULLAH EBN EL-ZUBAIR NOZHA HELIOPOLIS CAIRO Fax: 2435695 Tel: 2417338/2440375/2444997